
SSSSSSSSSSS YYY YYY SSSSSSSSSSS
SSSSSSSSSSS YYY YYY SSSSSSSSSSS
SSSSSSSSSSS YYY YYY SSSSSSSSSSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSSSSSSSS YYY SSSSSSSSS
SSSSSSSSS YYY SSSSSSSSS
SSSSSSSSS YYY SSSSSSSSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSSSSSSSSSS YYY SSSSSSSSSSS
SSSSSSSSSSS YYY SSSSSSSSSSS
SSSSSSSSSSS YYY SSSSSSSSSSS

FILEID**CMODSSDSP

```
  CCCCCCCC  MM      MM   OOOOOO    DDDDDDDD     SSSSSSSS   SSSSSSSS  DDDDDDDD     SSSSSSSS  PPPPPPPP
  CCCCCCCC  MM      MM   OOOOOO    DDDDDDDD     SSSSSSSS   SSSSSSSS  DDDDDDDD     SSSSSSSS  PPPPPPPP
CC          MMMM  MMMM OO      OO  DD      DD  SS         SS        DD      DD  SS         PP      PP
CC          MMMM  MMMM OO      OO  DD      DD  SS         SS        DD      DD  SS         PP      PP
CC          MM MM MM MM OO      OO  DD      DD  SS         SS        DD      DD  SS         PP      PP
CC          MM MM MM MM OO      OO  DD      DD  SS         SS        DD      DD  SS         PP      PP
CC          MM    MM MM OO      OO  DD      DD    SSSSSS     SSSSSS  DD      DD    SSSSSS   PPPPPPPP
CC          MM    MM MM OO      OO  DD      DD    SSSSSS     SSSSSS  DD      DD    SSSSSS   PPPPPPPP
CC          MM    MM MM OO      OO  DD      DD          SS         SS  DD      DD          SS PP
CC          MM    MM MM OO      OO  DD      DD          SS         SS  DD      DD          SS PP
CC          MM    MM MM OO      OO  DD      DD          SS         SS  DD      DD          SS PP
CC          MM    MM MM OO      OO  DD      DD          SS         SS  DD      DD          SS PP
  CCCCCCCC  MM    MM MM   OOOOOO    DDDDDDDD   SSSSSSSS   SSSSSSSS  DDDDDDDD   SSSSSSSS   PP
  CCCCCCCC  MM    MM MM   OOOOOO    DDDDDDDD   SSSSSSSS   SSSSSSSS  DDDDDDDD   SSSSSSSS   PP
```

```
MM      MM    AAAAAA    RRRRRRRR
MM      MM    AAAAAA    RRRRRRRR
MMMM  MMMM   AA    AA   RR     RR
MMMM  MMMM   AA    AA   RR     RR
MM MM MM MM  AA    AA   RR     RR
MM MM MM MM  AA    AA   RR     RR
MM    MM MM  AA    AA   RRRRRRRR
MM    MM MM  AA    AA   RRRRRRRR
MM    MM MM  AAAAAAAAAA RR   RR
MM    MM MM  AAAAAAAAAA RR   RR
MM    MM MM  AA    AA   RR    RR
MM    MM MM  AA    AA   RR    RR
MM    MM MM  AA    AA   RR     RR
MM    MM MM  AA    AA   RR     RR
```

```
        .NLIST  CND
        .IF     NDF,MPSWITCH
        .IF     NDF,RMSSWITCH
        .IF     NDF,LIBSWITCH
        .TITLE  CMODSSDSP - CHANGE MODE SYSTEM SERVICE DISPATCHER
        .IFF
        .IF     NDF,P1VSWITCH
        .TITLE  SYS$VECTOR - SYSTEM SERVICE VECTOR DEFINITIONS
        .IFF
        .TITLE  SYS$P1_VECTOR - P1 SYSTEM SERVICE VECTOR DEFINITIONS
        .ENDC
        .ENDC
        .IFF
        .TITLE  SYS$RMS_VECTOR - RMS SERVICE VECTOR DEFINITIONS
        .ENDC
        .IFF    ;MPSWITCH DEFINED
        .TITLE  MPCMOD - MULTIPROCESSING KERNEL SYS SRV DISPATCHER FOR SECONDARY
        .ENDC   ;MPSWITCH
        .IDENT  'V04-000'

;
;****************************************************************************
;*                                                                          *
;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
;*  ALL RIGHTS RESERVED.                                                    *
;*                                                                          *
;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE          *
;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER     *
;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY      *
;*  TRANSFERRED.                                                            *
;*                                                                          *
;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE    *
;*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT        *
;*  CORPORATION.                                                            *
;*                                                                          *
;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS     *
;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
;*                                                                          *
;*                                                                          *
;****************************************************************************

; D. N. CUTLER 22-JUN-76

;  MODIFIED BY:

;       V03-041 LJK0287         Lawrence J. Kenah       27-Jun-1984
;               Add R5 to entry mask for $CANEXH system service.

;       V03-040 LMP0239         L. Mark Pilant,         23-Apr-1984  9:21
;               Change $CHKPRO from an exec mode service to a kernel mode
;               service.  This was made necessary by the $CHKPRO (internal
;               entry point) interface change.
```

```
;       V03-039 MMD0250         Meg Dumont,     27-Feb-1984  17:49
;               Add support for $MTACCESS installation specific accessibility
;               routine
;
;       V03-038 DAS0001         David Solomon           20-Feb-1984
;               Implement new design for RMS echo SYS$INPUT to SYS$OUTPUT
;               (vs V03-019). Echo is now performed by a caller's mode AST
;               routine declared in RMS\RM$EXRMS. Change INCB/DECB of FAB/RAB
;               busy bit to BISB/BICB, now that we have room.
;
;       V03-037 SSA0004         Stan Amway              28-Dec-1983
;               For $SETPFM, changed number of parameters from 1 to 4
;               and changed entry mask to save R2-R11.
;
;       V03-036 TMK0002         Todd M. Katz            19-Nov-1983
;               The entry point for $ASCTOID can no longer be reached as a
;               branch destination from the executive mode dispatcher.
;               A temporary entry point (EXE$ASCTOID) has been placed within
;               this module, and a JMP is made from it to the real system
;               service entry point (EXE$$ASCTOID).
;
;               Also, change the entry mask for SYS$TRNLOG, so that R8 is
;               now saved.
;
;       V03-035 TMK0001         Todd M. Katz            22-Oct-1983
;               The entry points for $FINISH_RDB and $IDTOASC can no
;               longer be reached as branch destinations from the executive
;               mode dispatcher. Temporary entry points (EXE$FINISH_RDB and
;               EXE$IDTOASC) have been placed within this module, and from
;               each a JMP is made to the real system service entry points
;               (EXE$$FINISH_RDB and EXE$$IDTOASC).
;
;       V03-034 PRB0254         Paul Beck       15-Sep-1983  14:49
;               (1) Correct the way synchronous CJF services are defined.
;               (2) Define loadable RUF services.
;
;       V03-033 WMC0029         Wayne Cardoza           31-Aug-1983
;               Loadable services should not be unconditionally inhibited.
;               Add an alternate CHMx argument to LDBSRV.
;
;       V03-032 DWT0125         David W. Thiel          22-Aug-1983
;               Remove CHECKARGLIST and calls to same.
;
;       V03-031 MKL0167         Mary Kay Lyons          19-Aug-1983
;               Generate loadable service vector for CJF$GETCJI.
;
;       V03-030 KBT0578         Keith B. Thompson       8-Aug-1983
;               Add parameter to $FILESCAN
;
;       V03-029 RAS0178         Ron Schaefer            29-Jul-1983
;               Add code to detect the AST/non-AST RMS FAB/RAB race
;               condition where an RMS operation is initiated while
;               the user FAB/RAB is still waiting for completion of
;               previous operation.
;
;       V03-028 WMC0028         Wayne Cardoza           29-Jun-1983
```

```
;               Add CJF services.
;
;       V03-027 WMC0027         Wayne Cardoza           23-Jun-1983
;               Make old logical name services "all mode".
;               Changes to image activator vectors.
;
;       V03-026 JWH0222         Jeffrey W. Horn          2-May-1983
;               Add LDBSRV macro for vector definitions of loadable
;               services.
;
;       V03-025 DMW4035         DMWalp                  26-May-1983
;               Intergate new logical name structures.
;
;       V03-024 LMP0109         L. Mark Pilant,         28-Apr-1983  15:53
;               Make $CHKPRO an EXEC mode system service to allow examination
;               of various system data structures.
;
;       V03-024 RAS0147         Ron Schaefer            28-APR-1983
;               Add $FILESCAN. Add R8 and R9 to $SETPRN register mask.
;
;       V03-023 JLV0244         Jake VanNoy             27-APR-1983
;               Add $BRKTHRUW. Change $BRDCST to all mode service.
;               $BRDCST now uses $BRKTHRU to do real work.
;
;       V03-022 LMP0099         L. Mark Pilant,         13-Apr-1983  19:15
;               Add the $CHKPRO system service.
;
;       V03-021 ACG0319         Andrew C. Goldstein,    21-Mar-1983  13:51
;               Add $GRANTID and $REVOKID services
;
;       V03-020 JLV0234         Jake VanNoy              1-MAR-1983
;               Add $BRKTHRU service.
;
;       V03-019 RAS0120         Ron Schaefer            25-Feb-1983
;               Add support to echo SYS$INPUT to SYS$OUTPUT.
;               This involves examining the return code from RMS for $GET;
;               if the special status RMS$_ECHO (not returned to users)
;               is found, then create a RAB on the caller's stack and
;               execute a $PUT operation to echo the line.
;               A certain amount of RMS synchronization code was
;               shuffled around in order to make room for this.
;
;       V03-018 ACG0317         Andrew C. Goldstein,    22-Feb-1983  15:16
;               Fix off-by-one in kernel arg vector
;
;       V03-017 RSH0004         R. Scott Hanna          10-Feb-1983
;               Added $ASCTOID, $FINISH_RDB, and $IDTOASC to system service list
;
;       V03-016 RNG0016         Rod N. Gamache          1-Feb-1983
;               Added $GETLKI to system service list
;
;       V03-015 WMC0015         Wayne Cardoza           12-Jan-1983
;               Put back accidentally deleted space holder for RMS synchronization.
;
;       V03-014 DMW4023         DMWalp                  7-Jan-1983
;               Added $CRELNT, $CRELNM, $DELLNM and $TRNLNM
```

```
;       V03-013 KDM0033         Kathleen D. Morse       13-Dec-1982
;               Correct usage of an interlocked instruction to flush
;               the hardware cache queue.
;
;       V03-012 ROW0146         Ralph O. Weber          6-DEC-1982
;               Insert routine header comments for INHEXCP, CHECKARGLIST,
;               and EXE$CMODKRNLX (MPS$CMODKRNLX).  Move things around so
;               that EXE$CMODKRNL (MPS$CMODKRNL) header comments are near
;               EXE$CMODRKNL (MPS$CMODKRNL) and ASTEXIT comments are near
;               ASTEXIT.  Make basic kernal-mode .PSECT definition for Y$CMODK
;               or MP$CMOD1 immediately after executive mode code so that new
;               code can be inserted in a way that preserves routine headers,
;               conditional assembly, and .PSECT definitions.  Backout ROW145,
;               and in its place, correct conditional assembly of BGEQU 10$
;               after ACCVIO_RET so that it is assembled only for MPCMOD and
;               so that it is located before ACCVIO_RET.  Change PCB address
;               lookup at KERDSP in MPCMOD to use CTL$GL_PCB so that it works
;               correctly regardless of which processor executes it.
;
;       V03-011 ROW0145         Ralph O. Weber          29-NOV-1982
;               Move EXE$EXCPTN (and MPS$EXCPTN) to before ASTEXIT (or
;               MPS$ASTEXIT) in an attempt to make branch destinations in
;               EXE$CMODKRNL reach.
;
;       V03-010 KDM0030         Kathleen D. Morse       18-Nov-1982
;               Add logic to MPCMOD that allows the primary to execute
;               secondary-specific code, without turning into a secondary.
;
;       V03-009 MLJ0099         Martin L. Jack, 20-Oct-1982  19:42
;               Complete V03-002 by correcting mode and argument count of
;               $SNDJBC and removing temporary stubs.
;
;       V03-008 RIH0001         Richard I. Hustvedt     1-Jun-1982
;               Correct handling of AST queue by secondary processor to
;               avoid losing some AST notifications by incorrectly computing
;               PHD$B_ASTLVL.
;
;       V03-007 KDM0018         Kathleen D. Morse       30-Sep-1982
;               Add MPSWITCH logic to create a kernel system service
;               dispatcher for the secondary processor of an 11/782.
;
;       V03-006 STJ3028         Steven T. Jeffreys      26-Sep-1982
;               Added $ERAPAT system service vector.
;
;       V03-005 DWT0058         David Thiel             11-Aug-1982
;               Eliminate use of R2 while waiting for service
;               completion.
;
;       V03-004 JWH0001         Jeffrey W. Horn         26-Jul-1982
;               Add new RMS service, RMSRUHNDLR, an un-documented service
;               which acts as the Recovery Unit handler for RMS.
;
;       V03-003 PHL0102         Peter H. Lipman         16-Jul-1982
;               Fix new SYNCH logic to always return SS$_NORMAL,
;               not access IOSB if error from service, and return
```

```
;               error status from $SETEF if event flag cluster went away
;
;       V03-002 PHL0101         Peter H. Lipman         17-Jun-1982
;               Add $SYNCH system service and fix $QIOW and $ENQW to use the
;               new code for waiting for the combination of EFN and IOSB
;
;               Improve readability of conditionals.
;
;               Add $GETDVIW, $GETJPIW, $GETSYIW, $SNDJBC, $SNDJBCW, and
;               $UPDSECW.  All the waiting versions use common code.
;
;
; CHANGE MODE SYSTEM SERVICE DISPATCHER
;
; MACRO LIBRARY CALLS
;
        $ACBDEF                         ;DEFINE AST CONTROL BLOCK OFFSETS
        $CHFDEF                         ;DEFINE CONDITION HANDLING OFFSETS
        $ENQDEF                         ;DEFINE ENQ SYSTEM SERVICE ARGS
        $GETDVIDEF                      ;DEFINE GETDVI SYSTEM SERVICE ARGS
        $GETJPIDEF                      ;DEFINE GETJPI SYSTEM SERVICE ARGS
        $GETLKIDEF                      ;DEFINE GETLKI SYSTEM SERVICE ARGS
        $GETSYIDEF                      ;DEFINE GETSYI SYSTEM SERVICE ARGS
        $IPLDEF                         ;DEFINE INTERRUPT PRIORITY LEVELS
        .IF     DF,MPSWITCH
        $LCKDEF                         ;DEFINE INTERLOCK BITS
        .ENDC
        $PCBDEF                         ;DEFINE PCB OFFSETS
        $PHDDEF                         ;DEFINE PHD OFFSETS
        $PRDEF                          ;DEFINE PROCESSOR REGISTERS
        $PSLDEF                         ;DEFINE PROCESSOR STATUS FIELDS
        $RABDEF                         ;DEFINE RMS RAB FIELDS
        $RPBDEF                         ;DEFINE REBOOT PARAMETER BLOCK
        $QIODEF                         ;DEFINE QIO SYSTEM SERVICE ARGS
        $SGNDEF                         ;DEFINE SYSGEN PARAMETERS
        $SNDJBCDEF                      ;DEFINE SNDJBC SYSTEM SERVICE ARGS
        $SSDEF                          ;DEFINE SYSTEM STATUS VALUES
        $SYNCHDEF                       ;DEFINE SYNCH SYSTEM SERVICE ARGS
        $UPDSECDEF                      ;DEFINE UPDATE SECTION SYS SRV ARGS
;
; LOCAL EQUATES
;
        CAT0 =          1@0
        CAT7 =          1@7
        DEF_MASK =      CAT0!CAT7       ;INHIBIT FOR 'ALL' AND 'NOT EXIT'
        EXC_MASK =      CAT7            ;INHIBIT ONLY FOR 'ALL' CASE
;
; LOCAL MACROS
;
;       GSYSSRV - GENERATE SYSTEM SERVICE ENTRY VECTOR
;
;       GSYSSRV SRVNAME,MODE,NARG,REGISTERS,MASK,NOSYNC
;
;       WHERE:
```

```
;               SRVNAME - SERVICE NAME LESS ANY PREFIX (SYS$,EXE$,RMS$$)
;               MODE - MODE DESIGNATOR FOR SERVICE (K,E,ALL,R)
;               NARG - REQUIRED NUMBER OF ARGUMENTS
;               REGISTERS - REGISTER SAVE LIST
;               MASK - SERVICE INHIBIT MASK(BIT SET IN CAT INHIBITS)
;               NOSYNC - NON-ZERO IF RMS SYNCHRONIZATION CODE NOT TO BE INCLUDED
;
        .MACRO  GSYSSRV,SRVNAME,MODE,NARG,REGS,MASK=DEF_MASK,NOSYNC
        .IF     NDF,RMSSWITCH
        .IF     DF,LIBSWITCH
        .PSECT  $$$0000,QUAD
        .IFF
        .PSECT  $$$000,QUAD
        .ENDC
        .ALIGN  QUAD
        .IF DF  LIBSWITCH
SYS$'SRVNAME::
        .IFF
        .IF     NDF,MPSWITCH
        .WORD   ^M<REGS>
        SRVNAME'_MASK = ^M<REGS>
        .IFTF   ;MPSWITCH
        .IF B   NOSYNC
        SRV'MODE        SRVNAME,NARG,MASK
        .IFF
        SRV'MODE        SRVNAME,NARG,MASK,NOSYNC
        .ENDC
        .ENDC   ;MPSWITCH
        .IFT
        .BLKL   2
        .ENDC
        .IFF
        SRV'MODE        SRVNAME,NARG,MASK
        .ENDC
        .ENDM   GSYSSRV

;       GCOMPSRVB - GENERATE COMPOSITE SYSTEM SERVICE ENTRY VECTOR BEGIN
;
;       GCOMPSRVB  SRVNAME,REGISTER_MASK[,PREFIX]
;
;       WHERE:
;               SRVNAME - SERVICE NAME LESS ANY PREFIX (SYS$, EXE$)
;               REGISTER_MASK - SYMBOLIC REGISTER MASK, E.G QIO_MASK
;               PREFIX - IF SUPPLIED, THE PREFIX FOR THE SERVICE NAME.
;                        IF OMITTED, "SYS$" IS ASSUMED.
;
        .MACRO  GCOMPSRVB,SRVNAME,REGMSK,PREFIX=SYS$
        .IF     NDF,MPSWITCH
        .IF     NDF,RMSSWITCH
        .IF     DF,LIBSWITCH
        .PSECT  $$$0000,QUAD
        .IFF
        .PSECT  $$$000,QUAD
```

```
        .ENDC
        .ALIGN  QUAD
        .IF DF  LIBSWITCH
        .IIF    NOT_BLANK, <SRVNAME>,-
'PREFIX'SRVNAME::
        .IFF
        .ENABL  LSB
COMPSTRT=.
        .IIF    NOT_BLANK, <REGMSK>,-
        .WORD   <REGMSK>
        .ENDC
        .ENDC
        .ENDC   ;MPSWITCH
        .ENDM   GCOMPSRVB

;
;       GCOMPSRVE - GENERATE COMPOSITE SYSTEM SERVICE ENTRY VECTOR END
;
;       GCOMPSRVE       QUADWORDS
;
;       WHERE:
;               QUADWORDS - NUMBER OF QUADWORDS TO RESERVE FOR VECTOR
;

        .MACRO  GCOMPSRVE,QUADS
        .IF     NDF,MPSWITCH
        .IF     NDF,RMSSWITCH
        .IF     DF,LIBSWITCH
        .BLKQ   QUADS
        .IFF
COMPSIZE=.-COMPSTRT
        .IF     GE,QUADS*8-COMPSIZE
        .BLKB   QUADS*8-COMPSIZE
        .IFF
        .ERROR          ; VECTOR EXCEEDS ALLOCATED SIZE ;
        .ENDC
        .DSABL  LSB
        .ENDC
        .ENDC
        .ENDC   ;MPSWITCH
        .ENDM   GCOMPSRVE


;
;       SRVK - GENERATE ENTRY FOR KERNEL MODE SERVICE
;
;       SRVK    SRVNAME,NARG,MASK
;

        .MACRO  SRVK,SRVNAME,NARG,MASK
        .IF     NDF,RMSSWITCH
        .IF     DF,MPSWITCH
CMK$C_'SRVNAME==KCASCTR
        .IFF    ;MPSWITCH DEFINED
CMK$C_'SRVNAME=KCASCTR
        CHMK    #SRVNAME
```

```
        RET
        .PSECT  Y$CMODKN,BYTE
        .=KCASCTR
        ASSUME NARG LE 127
        .BYTE   NARG
        .PSECT  Y$CMODKX,BYTE
        .=KCASCTR
        .BYTE   MASK
        .PSECT  Y$CMODK,BYTE
        .SIGNED_WORD    EXE$'SRVNAME-KCASE+2
        .IFTF   ;MPSWITCH
SRVNAME=KCASCTR
KCASCTR=KCASCTR+1
        .ENDC   ;MPSWITCH
        .ENDC
        .ENDM   SRVK

;
;       SRVE - GENERATE ENTRY FOR EXECUTIVE MODE SERVICE
;

        .MACRO  SRVE,SRVNAME,NARG,MASK
        .IF     NDF,MPSWITCH
        .IF     NDF,RMSSWITCH
CME$C_'SRVNAME=ECASCTR
        CHME    #SRVNAME
        RET
        .PSECT  Y$CMODEN,BYTE
        .=ECASCTR
        ASSUME NARG LE 127
        .BYTE   NARG
        .PSECT  Y$CMODEX,BYTE
        .=ECASCTR
        .BYTE   MASK
        .PSECT  Y$CMODE,BYTE
        .SIGNED_WORD    EXE$'SRVNAME-ECASE+2
        .ENDC
SRVNAME=ECASCTR
ECASCTR=ECASCTR+1
        .ENDC   ;MPSWITCH
        .ENDM   SRVE
;
;
;    MACROS FOR GENERATING RMS SYSTEM VECTORS
;
        .MACRO  RMSSRV  SRVNAME NARG=1,REGS=<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>,-
                        MASK,NOSYNC=0
        GSYSSRV SRVNAME,R,NARG,<REGS>,MASK,NOSYNC
        .ENDM   RMSSRV
;
;  SRVR - GENERATE ENTRY FOR RMS SERVICE (EXEC MODE)
;
        .MACRO  SRVR    SRVNAME,NARG,MASK,NOSYNC
        .IF     NDF,MPSWITCH
        .IF     NDF,RMSSWITCH
CME$C_'SRVNAME=RCASCTR
```

```
        CHME    #SRVNAME
        .IF EQ NOSYNC
        .IIF GT <.+2-RMSSYNC>-127,-
RMSSYNC=RMSWBR                          ;RESET BRANCH DESTINATION
RMSWBR=.
        BRB     RMSSYNC
        .IFF
        RET
        .ENDC
        .PSECT  Y$CMODEN,BYTE
        .=RCASCTR
        ASSUME NARG LE 127
        .BYTE   NARG
        .PSECT  Y$CMODEX,BYTE
        .=RCASCTR
        .BYTE   MASK
        .IFF
        .PSECT  $$$RMSVEC,BYTE,NOWRT
        .SIGNED_WORD    RMS$'SRVNAME-RCASE+2
        .ENDC
SRVNAME=RCASCTR
RCASCTR=RCASCTR+1
        .ENDC   ;MPSWITCH
        .ENDM   SRVR

;
;       SRVALL - GENERATE ENTRY FOR ALL MODE SERVICE
;

        .MACRO  SRVALL,SRVNAME,NARG,MASK
        .IF     NDF,MPSWITCH
        .IF NDF,RMSSWITCH
        JMP     @#EXE$'SRVNAME+2
        .ENDC
        .ENDC   ;MPSWITCH
        .ENDM   SRVALL

        .PAGE
        .SBTTL  Macros for Loadable Services

;
;       LDBSRV - Generate Loadable Service Vector
;
;       LDBSRV  PREFIX,SRVNAME,MODE,REGS,SYN_EFN,SYN_IOSB,ALT_CHMX
;
;       Where:
;               PREFIX          - Prefix for system service vector entry point name
;               SRVNAME         - Service name less any prefix (SYS$,CJF$, etc.)
;               MODE            - Mode designator for service (K,E,ALL)
;               REGS            - Register save list
;               SYN_EFN         - Event flag argument number for $SYNCH
;               SYN_IOSB        - IOSB argument number for $SYNCH
;               ALT_CHMX        - Use same CHMx number as this service
;

        .MACRO  LDBSRV,PREFIX,SRVNAME,MODE,REGS,SYN_EFN,SYN_IOSB,ALT_CHMX
```

```
        .IF NDF,RMSSWITCH
        .IF NDF,MPSWITCH
            .IF DF,LIBSWITCH
                .PSECT  $$$0000,QUAD
                .ALIGN  QUAD
PREFIX''SRVNAME::
                .IF BLANK SYN_EFN
                    .BLKL    2
                .IFF
                    .BLKL    4
                .ENDC
            .IFF
                .PSECT  $$$000,QUAD
                .ALIGN  QUAD
                .WORD   ^M<REGS>
                SRVNAME' MASK = ^M<REGS>
                LVEC_'MODE PREFIX,SRVNAME,SYN_EFN,SYN_IOSB,ALT_CHMX
            .ENDC
        .ENDC   ; MPSWITCH
        .ENDC   ; RMSSWITCH
        .ENDM   LDBSRV

;
;       LVEC_K  - Kernel Mode Loadable System Service Vector
;
;       LVEC_K  PREFIX,SERVICE,EFN,IOSB
;

        .MACRO  LVEC_K,PREFIX,SERVICE,EFN,IOSB,ALT_CHMK
        .IF BLANK ALT_CHMK
            CMKSC_'SERVICE = PREFIX'KCASCTR
        .IFF
            CMKSC_'SERVICE = ALT_CHMK
        .ENDC
        CHMK #SERVICE
        .IF NOT_BLANK EFN
            PUSHL       #EFN
            PUSHL       #IOSB
            JMP         @#EXE$LDB_SYNCH
        .IFF
            RET
        .ENDC
        .IF BLANK ALT_CHMK
            SERVICE = PREFIX'KCASCTR
            PREFIX'KCASCTR = PREFIX'KCASCTR + 1
        .IFF
            SERVICE = ALT_CHMK
        .ENDC
        .ENDM   LVEC_K

;
;       LVEC_E  - Exec Mode Loadable System Service Vector
;
;       LVEC_E  PREFIX,SERVICE,EFN,IOSB
```

```
        .MACRO  LVEC_E,PREFIX,SERVICE,EFN,IOSB,ALT_CHME
        .IF BLANK ALT_CHME
            CME$C_'SERVICE = PREFIX'ECASCTR
        .IFF
            CME$C_'SERVICE = ALT_CHME
        .ENDC
        CHME    #SERVICE
        .IF NOT BLANK EFN
            PUSHL       #EFN
            PUSHL       #IOSB
            JMP         @#EXE$LDB_SYNCH
        .IFF
            RET
        .ENDC
        RET
        .IF BLANK ALT_CHME
            SERVICE = PREFIX'ECASCTR
            PREFIX'ECASCTR = PREFIX'ECASCTR + 1
        .IFF
            SERVICE = ALT_CHME
        .ENDC
        .ENDM   LVEC_E

;
;       LVEC_ALL - Mode of caller Loadable System Service Vector
;
        LVEC_ALL    PREFIX,SERVICE,EFN,IOSB

        .MACRO  LVEC_ALL,PREFIX,SERVICE,EFN,IOSB,ALT_CHMK
        JMP     @#EXE$'SERVICE
        .IF NOT BLANK EFN
            .ERROR      ; SYNCH NOT ALLOWED FOR ALL-MODE SERVICES
        .ENDC
        .ENDM   LVEC_ALL


        .IF     DF,RMSSWITCH
ECASCTR=0
        .IFF    ;RMSSWITCH
        .IF     NDF,LIBSWITCH
        .IF     NDF,MPSWITCH

;
; GLOBAL SYMBOLS
;

EXE$C_CMSTKSZ==4*5                      ;NUMBER OF LONGWORDS IN DISPATCH CALL FRAME
        .PAGE
        .SBTTL  CHANGE MODE TO EXECUTIVE DISPATCHER
;+
; EXE$CMODEXEC - CHANGE MODE TO EXECUTIVE DISPATCHER
;
; THIS ROUTINE IS AUTOMATICALLY VECTORED TO WHEN A CHANGE MODE TO EXECUTIVE
; INSTRUCTION IS EXECUTED. THE STATE OF THE STACK ON ENTRY IS:
;
; INPUTS:
```

```
;       00(SP) = CHANGE MODE PARAMETER CODE.
;       04(SP) = SAVED PC OF EXCEPTION.
;       08(SP) = SAVED PSL OF EXCEPTION.
;
;       00(AP) = NUMBER OF SYSTEM SERVICE CALL ARGUMENTS.
;       04(AP) = FIRST ARGUMENT.
;              .
;              .
;              .
;       4*N(AP) = N'TH ARGUMENT.
;
; OUTPUTS:
;
;       ***TBS***
;
; NOTE:
;
;       DISPATCH TO RMS ROUTINES ASSUMES THAT R3, R4, & R8 ARE NOT DESTROYED
;       BY THE THE SERVICE EXIT CODE FOR SUCCESSFUL RETURNS.
;-
        .PSECT  Y$CMODEX,BYTE           ;START OF THE MASK TABLE
B_EMASK:
        .PSECT  Y$CMODE,QUAD
EXACCVIO:                               ;CHANGE MODE TO EXEC ACCESS VIOLATION
        MOVL    SP,FP                   ;SET FP TO POINT TO CALL FRAME
        CMPW    R0,#RCASCTR             ;IS THIS A BUILTIN OR RMS FUNCTION?
        BGEQU   EXEDSP                  ;NO, NOT NECESSARILY ACCVIO
        BRW     ACCVIO_RET              ;
EXE$EXCPTNE::                           ;EXECMODE SYSTEM SERVICE EXCEPTION
        .WORD   0                       ;NULL ENTRY MASK
        BUG_CHECK SSRVEXCEPT            ;NON-FATAL EXCEPTION IF IN EXEC MODE
        MOVL    CHF$L_SIGARGLST(AP),R1  ;GET ADDRESS OF SIGNAL ARGUMENTS
        $EXIT_S CHF$L_SIG_NAME(R1)      ;AND EXIT WITH SIGNAL AS STATUS
EXINSARG:                               ;CHANGE MODE TO EXEC INSUFFICIENT ARGS
        CMPW    R0,#RCASCTR             ;IS THIS A BUILTIN OR RMS FUNCTION?
        BGEQU   EXEDSP                  ;NO, NOT NECESSARILY INSARG
        BRW     INSARG                  ;
        .ALIGN  QUAD
EXE$CMODEXECX::
        BICL3   8(SP),#PSL$M_CURMOD,R0  ;CHECK THE PREVIOUS MODE
        BNEQ    EXE$CMODEXEC            ;NO CHECK NEEDED FOR NON-USER MODE
        MOVZBL  (SP),R0                 ;PICK UP THE CHME CODE (MOD 256)
        BITB    W^B_EMASK[R0],@#CTL$GB_SSFILTER ;'AND' WITH THE INHIBIT MASK
        BEQL    EXE$CMODEXEC            ;THIS CODE IS ALLOWED
        MOVZWL  #SS$_INHCHME,R1         ;SET THE EXECPTION CODE
        BRW     INHEXCP                 ;AND REFLECT IT
        .ALIGN  QUAD
EXE$CMODEXEC::                          ;CHANGE MODE TO EXECUTIVE DISPATCH
                                        ;NOTE: MEMORY WRITING INSTRUCTIONS ARE
                                        ;CAREFULLY INTERLACED WITH REGISTER TO
                                        ;REGISTER OPERATIONS FOR SPEED.
        POPL    R0                      ;REMOVE CHANGE MODE PARAMETER FROM STACK
        PUSHAB  W^SRVEXIT               ;RETURN ADDRESS FOR CALL FRAME
        MOVZBL  R0,R1                   ;BOUND RANGE OF CHME CODE VALUES
```

```
        PUSHL   FP                      ;SAVE FP
        MOVZBL  W^B_EXECNARG[R1],R1     ;GET REQUIRED NUMBER OF ARGUMENTS
        PUSHL   AP                      ;SAVE AP
        MOVAL   @#4[R1],FP              ;CALCULATE LENGTH OF ARGUMENT LIST
        CLRQ    -(SP)                   ;PSW, REGISTER SAVE MASK FOR CALL FRAME
        IFNORD  FP,(AP),EXACCVIO        ;BR IF ARGLIST INACCESSIBLE
        MOVL    SP,FP                   ;SET FP TO POINT TO CALL FRAME
        CMPB    (AP),R1                 ;CHECK FOR REQUIRED NUMBER OF ARGUMENTS
        BLSSU   EXINSARG                ;INSUFFICIENT NUMBER OF ARGUMENTS
                                        ;  (R0 HAS CHME CODE)
EXEDSP: CASEW   R0,#0,S^#ECASMAX        ;DISPATCH TO PROPER SERVICE ROUTINE
ECASCTR=0                               ;START WITH 0 FOR CHME CODE
ECASE:                                  ;BASE OF CHME CASE TABLE
        .PSECT  Y$CMODEN,BYTE           ;REQUIRED NUMBER OF ARG TABLE
B_EXECNARG:                             ;DEFINE TABLE BASE

;
;       NOTE THAT THE OUT OF RANGE FALL THROUGH FROM THE CASEW FOLLOWS
;       MANY PAGES LATER IN THIS LISTING (SEE "ILLEGAL CHME" SUBTITLE).
;
        .IFTF   ;Regardless of MPSWITCH state



;
; Establish .PSECT for kernel-mode servicing code which follows
;
        .IFT    ;MPSWITCH not defined
        .PSECT  Y$CMODK,QUAD
        .IFF    ;MPSWITCH defined
        .PSECT  MP$CMOD1,QUAD
        .IFTF   ;Regardless of MPSWITCH state
        .PAGE
        .SBTTL  INHEXCP - Inhibited CHMK or CHME code handling

;+
; INHEXCP - Inhibited CHMK or CHME code handling
;
; FUNCTIONAL DESCRIPTION:
;
; When the ability to use specified system services is inhibited
; via the $SETSSF system service, this routine receives control
; when an attempt to execute an inhibited system service occurs.
;
        .IFT    ;MPSWITCH not defined
; INHEXCP  is called when no stack frame cleanup is required.
; INHEXCP1 is called when a call frame must be cleared from the stack.
;
; The result of this code is a signaled exception whose signal arguments are:
;       1) SS$_INHCHMK or SS$_INHCHME
;       2) the inhibited change mode code whose use was attempted
;       3) the offending PC and PSL
;
; INPUTS:
```

```
;
;    INHEXCP
;       R1      = SS error code (SS$_INHCHMK or SS$_INHCHME)
;       00(SP) = Change mode parameter code
;       04(SP) = Saved PC of exception
;       08(SP) = Saved PSL of exception
;
;    INHEXCP1
;       A change mode dispatcher call frame to be cleaned up
;       R0      = Change mode parameter code
;       R1      = SS error code (SS$_INHCHMK or SS$_INHCHME)
;       04(SP) = Saved PC of exception
;       08(SP) = Saved PSL of exception
;-
        .IFF    ;MPSWITCH defined
; The exception condition is returned to the primary processor for execption
; handling.
;
; INPUTS:
;
;       R1      = SS error code (SS$_INHCHMK or SS$_INHCHME)
;       00(SP) = Change mode parameter code
;       04(SP) = Saved PC of exception
;       08(SP) = Saved PSL of exception
;
; ENVIRONMENT:
;
;       This code executes on the secondary processor.
;       If interrupted at any point, may continue on the primary processor.
;-
        .IFT    ;MPSWITCH NOT DEFINED
INHEXCP1:
        MOVL    12(SP),FP               ;PICK UP THE OLD FP FROM FRAME
        ADDL    #5*4,SP                 ;CLEAN OFF THE FRAME
        PUSHL   R0                      ;RESTORE THE CHMX CODE
        .IFTF   ;MPSWITCH
INHEXCP:
        PUSHL   R1                      ;PUSH THE EXECPTION CODE
        PUSHL   #4                      ;PUSH THE NUMBER OF ARGUMENTS
        .IFT    ;MPSWITCH NOT DEFINED
        JMP     G^EXE$REFLECT           ;REFLECT THE EXCEPTION
        .IFF    ;MPSWITCH DEFINED
        IFPRIMARY <JMP G^EXE$REFLECT>   ;IF PRIMARY, THEN CONTINUE RIGHT ALONG
                                        ;IF SECONDARY, RETURN PROCESS TO PRIMARY
        EXTZV   #PSL$V_CURMOD,#PSL$S_CURMOD,16(SP),-(SP) ;CREATE PSL WITH PREV
        ROTL    #PSL$V_PRVMOD,(SP),(SP) ; MODE CORRECT AND CURRENT MODE = KERNEL
        PUSHAB  G^EXE$REFLECT           ;REFLECT THE EXCEPTION
        BRW     MPS$MPSCHED2            ; AND RETURN PROCESS TO PRIMARY
        .IFT    ;MPSWITCH NOT DEFINED
        .PAGE
        .SBTTL  ASTEXIT SYSTEM SERVICE
;+
; ASTEXIT - SERVICE TO EXIT AN ACTIVE AST AND ALLOW PENDING ASTS TO
;           BE DELIVERED.
```

```
;
; THIS SYSTEM SERVICE IS INVOKED WITH A CHMK #ASTEXIT NOT CONTAINED IN
; A STANDARD SYSTEM SERVICE VECTOR TO AVOID CLUTTERING THE STACK WITH AN
; ADDITIONAL CALL FRAME DURING AST EXIT PROCESSING.
;
; INPUTS:
;       NONE
;
; OUTPUTS:
;       PCB$B_ASTACT IS CLEARED FOR THE ISSUING MODE
;       PHD$B_ASTLVL IS SET TO THE ACCESS MODE OF THE NEXT PENDING AST, IF ANY.
;
;-


        .ALIGN  QUAD                    ;** THIS IS ADDED TO FIX
                                        ;** A BROKEN BRANCH INST. -
                                        ;** BEQL ASTEXIT IN EXE$CMODKRNL

ASTEXIT:                                ;EXIT ACTIVE AST
        EXTZV   #PSL$V_CURMOD,#PSL$S_CURMOD,4(SP),R0 ;GET PREVIOUS MODE
        PUSHL   R2                      ;SAVE R2 (PUSHR IS SLOWER!)
        PUSHL   R4                      ;SAVE R4
        MOVL    SCH$GL_CURPCB,R4        ;GET PCB CURRENT PCB ADDRESS
        SETIPL  #IPL$_ASTDEL            ;DISABLE KERNEL AST DELIVERY
        BBCCI   R0,PCB$B_ASTACT(R4),10$ ;CLEAR AST ACTIVE BIT FOR MODE
10$:    BSBW    SCH$NEWLVL              ;COMPUTE NEW AST LEVEL SETTING
        POPL    R4                      ;RESTORE R4
        POPL    R2                      ;RESTORE R2
        REI                             ;AND EXIT
        .IFF    ;MPSWITCH DEFINED
        .PAGE
        .SBTTL MPS$ASTEXIT - AST EXIT SYSTEM SERVICE FOR SECONDARY PROCESSOR
;+
; FUNCTIONAL DESCRIPTION:
;
; This is the AST exit system service routine for the secondary processor
; only.  It clears the AST active bit for the appropriate mode, in the
; process' PCB and then sets a new AST level (both in the PHD and the
; secondary's processor register).  Because an AST may be delivered by
; the primary while the secondary is executing this code, the routine
; is repeated until the head of the AST queue is stable.
;
; INPUTS:
;
;       (SP) - PC at time of interrupt
;      4(SP) - PSL at time of interrupt
;
; ENVIRONMENT:
;
;       Executes on the secondary processor.
;       If interrupted at any point, may continue on the primary processor.
;-
```

```
        .PSECT  MPS$CMOD2,BYTE
MPS$ASTEXIT:
        EXTZV   #PSL$V_CURMOD,#PSL$S_CURMOD,4(SP),R0 ; Get previous mode
        PUSHL   R4                      ; Save register
        PUSHL   R3                      ; Save register (This is faster)
        PUSHL   R2                      ; Save register (than a PUSHR.)
        MOVL    W^MPS$GL_CURPCB,R4      ; Get address of current process' PCB
        SETIPL  #IPL$_SYNCH             ; Disable system events
        BBCCI   R0,PCB$B_ASTACT(R4),10$ ; Clear AST active bit for this mode
10$:    MOVAL   PCB$L_ASTQFL(R4),R0     ; Get address of AST queue
        MOVL    #4,R2                   ; Assume null AST level
        MOVL    (R0),R1                 ; Get flink
        CMPL    R0,R1                   ; Is the queue empty?
        BEQL    20$                     ; Br on yes, set null AST level
        CLRL    R2                      ; Assume kernel mode
        ASSUME  ACB$V_KAST EQ 7
        TSTB    ACB$B_RMOD(R1)          ; Check for kernel AST
        BLSS    20$                     ; Br if not kernel AST
        BICB3   #^C<3>,ACB$B_RMOD(R1),R2 ; Get request mode
20$:    MOVL    PCB$L_PHD(R4),R3        ; Get address of PHD
        MTPR    R2,#PR$_ASTLVL          ; Set ASTLVL register
        MOVB    R2,PHD$B_ASTLVL(R3)     ; Set ASTLVL in PHD
        BBSSI   #LCK$V_INTERLOCK,W^MPS$GL_INTERLOCK,30$ ; Flush cache queue
30$:    CMPL    (R0),R1                 ; Has the head of the queue changed?
        BNEQ    10$                     ; Yes, repeat ASTLVL computation
        MOVQ    (SP)+,R2                ; Restore registers
        POPL    R4                      ; Restore register
        REI                             ; Return from interrupt
        .PSECT  MPS$CMOD1,QUAD
        .IFTF   ;MPSWITCH
        .PAGE
        .SBTTL  CHANGE MODE DETECTED ERROR HANDLING
;+
; ACCVIO - ACCESS VIOLATION DETECTED IN ARGUMENT LIST
; INSARG - INSUFFICIENT ARGUMENTS SUPPLIED FOR SERVICE
; SSFAIL - ABNORMAL STATUS RETURNED BY SERVICE ROUTINE
;
; THESE ROUTINES TAKE THE APPROPRIATE ACTION TO RETURN THE ERROR INDICATION
; TO THE ORIGINAL CALLER.
;
;-
        .ENABL  LSB
ACCVIO:                                 ;
        MOVL    SP,FP                   ;SET FRAME POINTER BEFORE RET
        CMPW    R0,#KCASCTR             ;IS THIS AN UNRECOGNIZED CODE?
        .IFF    ;MPSWITCH DEFINED
        BGEQU   10$                     ;YES, NOT NECESSARILY ACCVIO
        .IFT    ;MPSWITCH NOT DEFINED
        BGEQU   KERDSP                  ;YES, NOT NECESSARILY ACCVIO
ACCVIO_RET:                             ;
        .IFTF   ;MPSWITCH
        MOVZWL  #SS$_ACCVIO,R0          ;SET ACCESS VIOLATION
        RET                             ;

KINSARG:CMPW    R0,#KCASCTR             ;IS THIS AN UNRECOGNIZED CODE?
10$:    BGEQU   KERDSP                  ;YES, NOT NECESSARILY INSARG
```

```
        .IFT    ;MPSWITCH NOT DEFINED
INSARG: MOVZWL  #SS$_INSFARG,R0         ;SET INSUFFICIENT NUMBER OF ARGUMENTS
        .IFF    ;MPSWITCH DEFINED
        MOVZWL  #SS$_INSFARG,R0         ;SET INSUFFICIENT NUMBER OF ARGUMENTS
        .IFTF   ;MPSWITCH
        RET                             ;
SRVEXIT:                                ;SERVICE EXIT
        BLBC    R0,SSFAIL               ;BR IF ABNORMAL COMPLETION
SRVREI: REI                             ;
        .IFT    ;MPSWITCH NOT DEFINED
EXE$EXCPTN::                            ;SYSTEM SERVICE EXCEPTION
        .IFF    ;MPSWITCH DEFINED
MPS$EXCPTN::                            ;SYSTEM SERVICE EXCEPTION
        .IFTF   ;MPSWITCH
        .WORD   0                       ;ENTRY MASK
        .IFT    ;MPSWITCH NOT DEFINED
        BUG_CHECK SSRVEXCEPT,FATAL      ;UNEXPECTED SYSTEM SERVICE EXCEPTION
        .IFF    ;MPSWITCH DEFINED
        SECBUG_CHECK SSRVEXCEPT,FATAL   ;UNEXPECTED SYSTEM SERVICE EXCEPTION
        .IFTF   ;MPSWITCH
SSFAIL: BITL    #7,R0                   ;TEST SEVERITY FIELD
        BEQL    SRVREI                  ;IF EQL WARNING
        BRW     SSFAILMAIN              ;GOTO MAIN SSFAIL LOGIC
        .DSABL  LSB
        .PAGE
        .SBTTL  Filtered Change Mode to Kernel Dispatcher
;+
;
        .IFT    ;MPSWITCH not defined
; EXE$CMODKRNLX - Filtered Change Mode to Kernel Dispatcher
        .IFF    ;MPSWITCH defined
; MPS$CMODKRNLX - Secondary Filtered Change Mode to Kernel Dispatcher
        .IFTF   ;Regardless of MPSWITCH state
;
; When inhibiting of user mode system service calls has been enabled via the
        .IFT    ;MPSWITCH not defined
; SSINHIBIT SYSGEN parameter, this routine -- not EXE$CMODKRNLX -- is called
        .IFF    ;MPSWITCH defined
; SSINHIBIT SYSGEN parameter, this routine -- not MPS$CMODKRNLX -- is called
        .IFTF   ;Regardless of MPSWITCH state
; whenever a CHMK instruction is executed.  The state of the stack on entry
; is:
;
; INPUTS:
;
;       00(SP) = CHANGE MODE PARAMETER CODE.
;       04(SP) = SAVED PC OF EXCEPTION.
;       08(SP) = SAVED PSL OF EXCEPTION.
;
;       00(AP) = NUMBER OF SYSTEM SERVICE CALL ARGUMENTS.
;       04(AP) = FIRST ARGUMENT.
;              .
;              .
;              .
;       4*N(AP) = N'TH ARGUMENT.
;
```

```
; OUTPUTS:
;
;       THE APPROPRIATE KERNEL MODE SYSTEM SERVICE IS INVOKED.
;-

        .IFT    ;MPSWITCH not defined
        .PSECT  Y$CMODKX,BYTE           ;START OF THE MASK TABLE
SYS$GB_KMASK::
        .BYTE   0                       ;ALLOW FOR ASTEXIT (CHMK #0)!!!
        .PSECT  Y$CMODK,QUAD
        .IFF    ;MPSWITCH defined
        .PSECT  MPS$CMOD1,QUAD
        .IFTF   ;Regardless of MPSWITCH state

        .ALIGN  QUAD
        .IFT    ;MPSWITCH not defined
EXE$CMODKRNLX::
        .IFF    ;MPSWITCH defined
MPS$CMODKRNLX::
        .IFTF   ;Regardless of MPSWITCH state
        BICL3   8(SP),#PSL$M_CURMOD,R0  ;CHECK THE PREVIOUS MODE
        .IFT    ;MPSWITCH NOT DEFINED
        BNEQ    EXE$CMODKRNL            ;NO CHECK NEEDED FOR NON-USER MODE
        .IFF    ;MPSWITCH DEFINED
        BNEQ    W^MPS$CMODKRNL          ;NO CHECK NEEDED FOR NON-USER MODE
        .IFTF   ;MPSWITCH
        MOVZBL  (SP),R0                 ;PICK UP THE CHMK CODE
        .IFT    ;MPSWITCH NOT DEFINED
        BITB    W^SYS$GB_KMASK[R0],G^CTL$GB_SSFILTER ;'AND' WITH INHIBIT MASK
        BEQL    EXE$CMODKRNL            ;THIS CODE IS ALLOWED
        .IFF    ;MPSWITCH DEFINED
        BITB    G^SYS$GB_KMASK[R0],G^CTL$GB_SSFILTER ;'AND' WITH INHIBIT MASK
        BEQL    W^MPS$CMODKRNL          ;THIS CODE IS ALLOWED
        .IFTF   ;MPSWITCH
        MOVZWL  #SS$_INHCHMK,R1         ;SET THE EXECPTION CODE
        BRW     INHEXCP                 ;AND REFLECT IT

        .PAGE
        .SBTTL  CHANGE MODE TO KERNEL DISPATCHER
;+
        .IFT    ;MPSWITCH NOT DEFINED
; EXE$CMODKRNL - CHANGE MODE TO KERNEL DISPATCHER
        .IFF    ;MPSWITCH DEFINED
; MPS$CMODKRNL - SECONDARY CHANGE MODE TO KERNEL DISPATCHER
        .IFTF   ;MPSWITCH
;
; THIS ROUTINE IS AUTOMATICALLY VECTORED TO WHEN A CHANGE MODE TO KERNEL
; INSTRUCTION IS EXECUTED. THE STATE OF THE STACK ON ENTRY IS:
;
; INPUTS:
;
;       00(SP) = CHANGE MODE PARAMETER CODE.
;       04(SP) = SAVED PC OF EXCEPTION.
;       08(SP) = SAVED PSL OF EXCEPTION.
;
;       00(AP) = NUMBER OF SYSTEM SERVICE CALL ARGUMENTS.
```

```
;       04(AP) = FIRST ARGUMENT.
;              .
;              .
;              .
;       4*N(AP) = N'TH ARGUMENT.
;
; OUTPUTS:
;
;       THE APPROPRIATE KERNEL MODE SYSTEM SERVICE IS INVOKED.
;-
        .ALIGN  QUAD
        .IFT    ;MPSWITCH NOT DEFINED
EXE$CMODKRNL::                          ;CHANGE MODE TO KERNEL DISPATCH
        .IFF    ;MPSWITCH DEFINED
MPS$CMODKRNL::                          ;2NDARY CHANGE MODE TO KERNEL DISPATCH
        .IFTF   ;MPSWITCH
                                        ;NOTE: MEMORY WRITING INSTRUCTIONS ARE
                                        ;CAREFULLY INTERLACED WITH REGISTER
                                        ;INSTRUCTIONS FOR SPEED.

        .IF     DF,MPPFMSWT
        PUSHL   #^X40                   ;OFFSET INTO SCB
        BSBW    W^MPS$PFM_UNEXP         ;COUNT WHICH SYSTEM SERVICE IS EXECUTED
        ADDL    #4,SP                   ;CLEAN OFF SCB OFFSET
        .ENDC
        POPL    R0                      ;REMOVE CHANGE MODE PARAMETER FROM STACK
        .IFT    ;MPSWITCH NOT DEFINED
        BEQL    ASTEXIT                 ;IF ZERO, AST EXIT SYSTEM SERVICE
        .IFF    ;MPSWITCH DEFINED
        BEQL    ASTEXIT                 ;IF ZERO, AST EXIT SYSTEM SERVICE
        .IFTF   ;MPSWITCH
        PUSHAB  B^SRVEXIT               ;RETURN ADDRESS
        MOVZBL  R0,R1                   ;BOUND RANGE OF CHMK CODES TO 0,255
                                        ;AND 256 BYTES ACCESSIBLE FROM B_KRNLNARG
        PUSHL   FP                      ;SAVE FP
        .IFT    ;MPSWITCH NOT DEFINED
        MOVZBL  W^SYS$GB_KRNLNARG[R1],R1 ;GET NUMBER OF REQUIRED ARGUMENTS
        .IFF    ;MPSWITCH DEFINED
        MOVZBL  G^SYS$GB_KRNLNARG[R1],R1 ;GET NUMBER OF REQUIRED ARGUMENTS
        .IFTF   ;MPSWITCH
        PUSHL   AP                      ;SAVE AP
        MOVAL   @#4[R1],FP              ;CALCULATE LENGTH OF ARGUMENT LIST
        CLRQ    -(SP)                   ;PSW AND REGISTER SAVE MASK
        .IFT    ;MPSWITCH NOT DEFINED
        IFNORD  FP,(AP),ACCVIO          ;DECLARE ACCESS VIOLATION
        .IFF    ;MPSWITCH DEFINED
        IFNORD  FP,(AP),ACCVIO1         ;DECLARE ACCESS VIOLATION
        .IFTF   ;MPSWITCH
        MOVL    SP,FP                   ;SET FRAME POINTER FOR CALL FRAME
        CMPB    (AP),R1                 ;CHECK FOR REQUIRED NUMBER OF ARGS
        .IFT    ;MPSWITCH NOT DEFINED
        BLSSU   KINSARG                 ;IF LSSU, INSUFFICIENT ARGUMENTS
KERDSP: MOVL    G^SCH$GL_CURPCB,R4      ;GET CURRENT PROCESS PCB ADDRESS
        CASEW   R0,#1,#KCASMAX          ;DISPATCH TO PROPER SERVICE ROUTINE
        .IFF    ;MPSWITCH DEFINED
```

```
        BLSSU   KINSARG1                ;IF LSSU, INSUFFICIENT ARGUMENTS
KERDSP: MOVL    G^CTL$GL_PCB,R4         ;GET CURRENT PROCESS PCB ADDRESS
        CMPW    R0,#WAITFR              ;IS THIS THE WAITFR SYSTEM SERVICE?
        BEQL    MPS$WAITFR1             ;BR ON YES, EXECUTE SYS SRV ON SECONDARY
        CMPW    R0,#WFLAND              ;IS THIS THE WFLAND SYSTEM SERVICE?
        BEQL    MPS$WFLAND1             ;BR ON YES, EXECUTE SYS SRV ON SECONDARY
        CMPW    R0,#WFLOR               ;IS THIS THE WFLOR SYSTEM SERVICE?
        BEQL    MPS$WFLOR1              ;BR ON YES, EXECUTE SYS SRV ON SECONDARY
        ADDL    #8,SP                   ;CLEAN OFF PSW AND REG SAVE MASK
        POPL    AP                      ;RESTORE AP
        POPL    FP                      ;RESTORE FP
        MOVL    R0,(SP)                 ;REPLACE CHMK ON STACK OVER RET ADR
        IFPRIMARY <JMP G^EXE$CMODKRNL>  ;IF PRIMARY, THEN CONTINUE RIGHT ALONG
                                        ;IF SECONDARY, RETURN PROCESS TO PRIMARY
        EXTZV   #PSL$V_CURMOD,#PSL$S_CURMOD,8(SP),-(SP) ;CREATE PSL WITH PREV
        ROTL    #PSL$V_PRVMOD,(SP),(SP) ; MODE CORRECT AND CURRENT MODE = KERNEL
        PUSHAB  G^EXE$CMODKRNL          ;EXECUTE THE SERVICE ON PRIMARY
        BRW     MPS$MPSCHED2            ; AND RETURN PROCESS TO PRIMARY

ASTEXIT:
        BRB     MPS$ASTEXIT             ;BRANCH ASSIST
ACCVIO1:
        BRW     ACCVIO                  ;BRANCH ASSIST
KINSARG1:
        BRW     KINSARG                 ;BRANCH ASSIST

;
; BRANCH ASSISTS TO REACH SYSTEM SERVICES.
;
MPS$WAITFR1:
        BRW     MPS$WAITFR+2            ;BRANCH ASSIST (PAST REG SAVE MASK)
MPS$WFLAND1:
        BRW     MPS$WFLAND+2            ;BRANCH ASSIST (PAST REG SAVE MASK)
MPS$WFLOR1:
        BRW     MPS$WFLOR+2             ;BRANCH ASSIST (PAST REG SAVE MASK)
        .IFTF   ;MPSWITCH
KCASE:                                  ;BASE OF CHMK CASE TABLE
KCASCTR=1                               ;CHMK CODES START AT 1
        .IFT    ;MPSWITCH NOT DEFINED
        .PSECT  Y$CMODKN,BYTE           ;REQUIRED NUMBER OF ARG TABLE
SYS$GB_KRNLNARG==.
        .BYTE   0                       ;ENTRY FOR CODE ZERO
        .ENDC   ;MPSWITCH
        .ENDC   ;LIBSWITCH
        .IFF    ;RMSSWITCH
        .IF     NDF,MPSWITCH
        .PAGE
        .SBTTL  SYSTEM SERVICE VECTOR DEFINITION
;
;       DEFINE ALL SYSTEM SERVICE VECTOR POSITIONS
;
;
        .IF     NDF,LIBSWITCH           ;REAL PSECT IF NOT LIBRARY
        .PSECT  $$$000,QUAD
```

```
        .IFF                            ;OTHERWISE ABS PSECT
        .PSECT  $$$0000,QUAD,ABS
        .IF     NDF,P1VSWITCH
.=^X80000000                            ;BIASED AT THE START OF SYSEM SPACE
        .IFF    ;P1VSWITCH
.=^X7FFEDE00                            ;BIASED IN P1 SPACE
        .ENDC   ;P1VSWITCH
        .ENDC   ;LIBSWITCH
        .ENDC   ;MPSWITCH
        .IFF    ;RMSSWITCH
        .IF     NDF,MPSWITCH
VECBASE:                                ;VECTOR AREA BASE

;
;       QIO AND WAIT COMPOSITE SERVICE
;
;       THE QIO AND WAITFR COMPOSITE SERVICE OCCUPIES THE FIRST TWO
;       SYSTEM SERVICE VECTOR POSITIONS.  IT IS CONSTRUCTED BY
;       FROM TWO DISCRETE CHMK INSTRUCTIONS, ONE PERFORMING THE QIO
;       AND THE OTHER PERFORMING THE WAITFR, WHICH RELY UPON THE
;       COMPATIBLE ARGUMENT LISTS OF THESE TWO SERVICES. WAITFR HAS
;       A SINGLE ARGUMENT, THE EVENT FLAG, WHICH IS THE FIRST ARGUMENT
;       IN THE QIO ARGUMENT LIST.
;

        GCOMPSRVB QIOW,-                ;QIO AND WAIT
                <QIO_MASK ! WAITFR_MASK ! CLREF_MASK ! SETEF_MASK>
        .IF     NDF,LIBSWITCH
        CHMK    #QIO                    ;ISSUE QI/O
        BLBC    R0,QIOW_RET             ;DON'T WAIT IF ERROR QUEUEING REQUEST
        PUSHL   QIO$_IOSB(AP)           ;FETCH IOSB ADDRESS IF SPECIFIED
        BRW     QIO_ENQ_SYNCH           ;USE COMMON QIOW, ENQW SYNCH CODE
        .ENDC   ;LIBSWITCH
        GCOMPSRVE       2               ;RESERVE 2 QUADWORDS FOR VECTOR
        .ENDC   ;MPSWITCH
        .IFF    ;RMSSWITCH
        .IF     NDF,MPSWITCH

;
;  CONDITION HANDLER DISPATCH VECTOR
;
;  THE FOLLOWING VECTOR IS INCLUDED IN THE SYSTEM VECTOR SPACE SO THAT BOTH
;  HARDWARE-DETECTED (EXCEPTIONS) AND SOFTWARE-DETECTED (SIGNALS) CONDITIONS
;  CAN BE DISPATCHED FROM THE SAME CALL INSTRUCTION. THIS IS NECESSARY SO
;  THAT THE STACK SEARCH ALGORITHM AND THE UNWIND SYSTEM SERVICE CAN DETECT
;  AND PROPERLY PROCESS MULTIPLE ACTIVE SIGNALS AND/OR EXCEPTIONS.
;

        .ALIGN  QUAD
        .IF DF  LIBSWITCH
        .IF DF  P1VSWITCH               ;DON'T PUT IN P1
SYS$CALL_HANDL == . - ^X7FFEDE00 + ^X80000000
        .IFF    ;P1VSWITCH
SYS$CALL_HANDL::                        ;CONDITION HANDLER DISPATCH
        .ENDC   ;P1VSWITCH
        .IFF    ;LIBSWITCH
```

```
        CALLG   4(SP),(R1)              ;CALL CONDITION HANDLER
        RSB                             ;
;
; RET INSTRUCTION FOR QIOW ABOVE
;
QIOW_RET:
        RET
        .IFT    ;LIBSWITCH
        .BLKQ   1                       ;RESERVE SPACE
        .ENDC   ;LIBSWITCH
        .ENDC   ;MPSWITCH
        .IFF    ;RMSSWITCH
        .IF     NDF,MPSWITCH

;
; COMMAND INTERPRETER DISPATCH VECTOR
;
; THE FOLLOWING VECTOR IS INCLUDED IN THE SYSTEM VECTOR SPACE SO THAT DIRECT
; CALLS CAN BE MADE TO THE CURRENT COMMAND INTERPRETER WITHOUT HAVING TO KNOW
; THE ADDRESS OF ITS SERVICE ROUTINE.
;
        .ALIGN  QUAD
        .IF DF  LIBSWITCH
SYS$CLI::                               ;COMMAND INTERPRETER DISPATCH
        .IFF    ;LIBSWITCH
        .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;SAVE R2-R11
        JMP     CLIJMP                  ;INDIRECT DISPATCH TO CURRENT COMMAND INTERPRETER
        .IFT    ;LIBSWITCH
        .BLKQ   1                       ;RESERVE SPACE
        .ENDC   ;LIBSWITCH
        .IFF    ;RMSSWITCH
        .ALIGN  QUAD
        .ENDC   ;RMSSWITCH
        .ENDC   ;MPSWITCH
        .PAGE
;
;       DEFINE REMAINING SERVICES
;

GSYSSRV ADJSTK,K,3,-                    ;ADJUST OUTER MODE STACK POINTER
        <R2,R3,R4,R5,R6>,-              ;REGISTERS R2-R6
        EXC MASK                        ;EXCEPTION MASK
GSYSSRV ADJWSL,K,2,-                    ;ADJUST WORKING SET LIMIT
        <R2,R3,R4,R5>                   ;REGISTERS R2-R5
GSYSSRV ALCDNP,K,4,-                    ;ALLOCATE DIAGNOSTIC PAGE
        <R2,R3,R4,R5,R6,R7>             ;REGISTERS R2-R7
GSYSSRV ALLOC,K,4,-                     ;ALLOCATE DEVICE
        <R2,R3,R4,R5,R6>                ;REGISTERS R2-R6
GSYSSRV ASCEFC,K,4,-                    ;ASSOCIATE COMMON EVENT FLAG CLUSTER
        <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;REGISTERS R2-R11
GSYSSRV ASCTIM,ALL,3,-                  ;CONVERT TO ASCII TIME
        <R2,R3,R4,R5,R6>                ;REGISTERS R2-R6
GSYSSRV ASSIGN,K,4,-                    ;ASSIGN I/O CHANNEL
        <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;REGISTERS R2-R11
GSYSSRV BINTIM,ALL,2,-                  ;CONVERT TO BINARY TIME
```

```
        <R2,R3,R4,R5,R6,R7,R8>  ;REGISTERS R2-R8
GSYSSRV CANCEL,K,1,-            ;CANCEL I/O ON CHANNEL
        <R2,R3,R4,R5,R6,R7,R8>  ;REGISTERS R2-R8
GSYSSRV CANTIM,K,2,-            ;CANCEL TIMER REQUEST
        <R2,R3,R4,R5>           ;REGISTERS R2-R5
GSYSSRV CANWAK,K,2,-            ;CANCEL WAKE UP REQUESTS
        <R2,R3,R4,R5>           ;REGISTERS R2-R5
GSYSSRV CRMPSC,K,12,-           ;CREATE AND MAP SECTION
        <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;REGISTERS R2-R11
GSYSSRV CLRPAR,K,2,-            ;CLEAR HARD PARITY ERROR
        <R2,R3,R4,R5>           ;REGISTERS R2-R5
GSYSSRV CMEXEC,E,2,-            ;CHANGE MODE TO EXECUTIVE
        <R4>                    ;REGISTER R4
GSYSSRV CMKRNL,K,2,-            ;CHANGE MODE TO KERNEL
        <R4>                    ;REGISTER R4
GSYSSRV CLREF,K,1,-             ;CLEAR EVENT FLAG
        <R2,R3,R4,R5>           ;REGISTERS R2-R5. SEE WAITFR COMMENTS.
GSYSSRV CNTREG,K,4,-            ;CONTRACT REGION
        <R2,R3,R4,R5,R6,R7>     ;REGISTERS R2-R7
GSYSSRV GETPTI,K,5,-            ;GET PAGE TABLE INFORMATION
        <R2,R3,R4,R5,R6,R7,R8,R9,R10> ;REGISTERS R2-R10
GSYSSRV CRELOG,ALL,4,-          ;CREATE LOGICAL NAME
        <R2,R3,R4,R5,R6,R7,R8>  ;REGISTERS R2-R8
GSYSSRV CREMBX,K,7,-            ;CREATE MAILBOX
        <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;REGISTERS R2-R11
GSYSSRV CREPRC,K,12,-           ;CREATE PROCESS
        <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;REGISTERS R2-R11
GSYSSRV CRETVA,K,3,-            ;CREATE VIRTUAL ADDRESS
        <R2,R3,R4,R5,R6,R7,R8>,-;REGISTERS R2-R8
        EXC_MASK                ;EXCEPTION MASK
GSYSSRV DACEFC,K,1,-            ;DISASSOCIATE EVENT FLAG CLUSTER
        <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;REGISTERS R2-R11
GSYSSRV DALLOC,K,2,-            ;DEALLOCATE DEVICE
        <R2,R3,R4,R5,R8>        ;REGISTERS R2-R5,R8
GSYSSRV DASSGN,K,1,-            ;DEASSIGN I/O CHANNEL
        <R2,R3,R4,R5,R6,R7,R8>  ;REGISTERS R2-R8
GSYSSRV DCLAST,K,3,-            ;DECLARE AST SYSTEM SERVICE
        <R2,R3,R4,R5>           ;REGISTERS R2-R5
GSYSSRV DCLEXH,K,1,-            ;DECLARE EXIT HANDLER
        <R2,R3,R4>              ;REGISTERS R2-R4
GSYSSRV DELLOG,ALL,3,-          ;DELETE LOGICAL NAME
        <R2,R3,R4,R5,R6,R7,R8>  ;REGISTERS R2-R8
GSYSSRV DELMBX,K,1,-            ;DELETE MAILBOX
        <R2,R3,R4,R5>           ;REGISTERS R2-R5
GSYSSRV DELPRC,K,2,-            ;DELETE PROCESS
        <R2,R3,R4,R5,R6,R7>     ;REGISTERS R2-R5
GSYSSRV DELTVA,K,3,-            ;DELETE VIRTUAL ADDRESS
        <R2,R3,R4,R5,R6,R7>,-   ;REGISTERS R2-R7
        EXC_MASK                ;EXCEPTION MASK
GSYSSRV DGBLSC,K,3,-            ;DELETE GLOBAL SECTION
        <R2,R3,R4,R5,R6,R7,R8,R9,R10> ;REGISTERS R2-R10
GSYSSRV DLCDNP,K,2,-            ;DEALLOCATE DIAGNOSTIC PAGE
        <R2,R3,R4,R5,R6,R7>     ;REGISTERS R2-R7
GSYSSRV DLCEFC,K,1,-            ;DELETE COMMON EVENT CLUSTER
        <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;REGISTERS R2-R11
GSYSSRV UPDSEC,K,8,-            ;UPDATE SECTION FILE
```

```
        <R2,R3,R4,R5,R6,R7,R8>  ;R2-R8
GSYSSRV SNDERR,K,1,-            ;SEND MSG TO ERROR LOGGER
        <R2,R3,R4,R5>           ;REGISTERS R2-R5
GSYSSRV EXIT,K,1,-              ;IMAGE EXIT
        <R4>,0                  ;REGISTER R4, ALWAYS ALLOWED!
GSYSSRV EXPREG,K,4,-            ;EXPAND PROGRAM REGION
        <R2,R3,R4,R5,R6,R7,R8>  ;REGISTERS R2-R8
GSYSSRV FAO,ALL,0,-             ;FORMAT ASCII OUTPUT
        <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; REGISTERS R2-R11
GSYSSRV FAOL,ALL,0,-            ;FORMAT ASCII OUTPUT WITH VALUE LIST
        <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;REGISTERS R2-R11
GSYSSRV FORCEX,K,3,-            ;FORCE EXIT
        <R2,R3,R4,R5>           ;REGISTERS R2-R5
GSYSSRV IMGSTA,ALL,6,-          ;IMAGE STARTUP
        <>                      ;REGISTERS NONE
GSYSSRV SNDJBC,E,7,-            ;SEND TO JOB CONTROLLER
        <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;REGISTERS R2-R11
GSYSSRV GETTIM,E,1,-            ;GET TIME
        <>                      ;NO REGISTERS
GCOMPSRVB UPDSECW,-             ;UPDATE SECTION AND WAIT
        <UPDSEC_MASK ! GETJPI_SYNCH_MASK>
.IF     NDF,MPSWITCH
.IF     NDF,RMSSWITCH
.IF     NDF,LIBSWITCH
JMP     @#EXE$UPDSECW
.ENDC   ;LIBSWITCH
.ENDC   ;RMSSWITCH
.ENDC   ;MPSWITCH
GCOMPSRVE       1
GSYSSRV HIBER,K,0,-             ;HIBERNATE
        <R2,R3,R4,R5>           ;REGISTERS R2-R5
GSYSSRV IMGACT,E,8,-            ;IMAGE ACTIVATION
        <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;REGISTERS R2-R11
GSYSSRV LCKPAG,K,3,-            ;LOCK PAGE IN MEMORY
        <R2,R3,R4,R5,R6,R7,R8>  ;REGISTERS R2-R8
GSYSSRV LKWSET,K,3,-            ;LOCK PAGES IN WORKING SET
        <R2,R3,R4,R5,R6,R7,R8>  ;REGISTERS R2-R8
GSYSSRV MGBLSC,K,7,-            ;MAP GLOBAL SECTION
        <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;REGISTERS R2-R11
GSYSSRV PURGWS,K,1,-            ;PURGE WORKING SET
        <R2,R3,R4,R5,R6,R7,R8>  ;R2-R8
GSYSSRV NUMTIM,E,2,-            ;CONVERT TIME TO NUMERIC
        <R2,R3,R4,R5,R6,R7>     ;REGISTERS R2-R7
GSYSSRV SNDOPR,E,2,-            ;SEND MSG TO OPERATOR
        <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>       ;REGISTERS R2-R11
GSYSSRV QIO,K,12,-              ;QUEUE I/O REQUEST
        <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;REGISTERS R2-R11
GSYSSRV READEF,K,2,-            ;READ EVENT FLAG
        <R2,R3,R4,R5>           ;REGISTERS R2-R5
GSYSSRV RESUME,K,2,-            ;RESUME PROCESS
        <R2,R3,R4,R5>           ;REGISTERS R2-R5
GSYSSRV RUNDWN,K,1,-            ;RUNDOWN
        <R2,R3,R4,R5,R6,R7>     ;REGISTERS R2-R7
GSYSSRV SNDSMB,E,2,-            ;SEND MSG TO SYMBIONT MANAGER
        <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>       ;REGISTERS R2-R11
GSYSSRV SCHDWK,K,4,-            ;SCHEDULE WAKEUP
```

```
                <R2,R3,R4,R5,R6,R7,R8,R9> ;REGISTERS R2-R9
        GSYSSRV SETAST,K,1,-              ;SET AST ENABLE SERVICE
                <R2,R3,R4,R5>             ;REGISTERS R2-R5
        GSYSSRV SETEF,K,1,-               ;SET EVENT FLAG
                <R2,R3,R4,R5>             ;REGISTERS R2-R5. SEE WAITFR COMMENTS.
        GSYSSRV SETEXV,K,4,-              ;SET EXCEPTION VECTOR
                <R2,R3,R4,R5>             ;REGISTERS R2-R5
        GSYSSRV SETPRN,K,1,-              ;SET PROCESS  NAME
                <R2,R3,R4,R5,R6,R7,R8,R9> ;REGISTERS R2-R9
        GSYSSRV SETPRA,K,2,-              ;SET POWER RECOVERY AST
                <R2,R3,R4,R5>             ;REGISTERS R2-R5
        GSYSSRV SETIMR,K,4,-              ;SET TIMER
                <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;REGISTERS R2-R11
        GSYSSRV SETPRI,K,4,-              ;SET PROCESS PRIORITY
                <R2,R3,R4,R5>             ;REGISTERS R2-R5
        GSYSSRV SETPRT,K,5,-              ;SET PAGE PROTECTION
                <R2,R3,R4,R5,R6,R7,R8,R9> ;REGISTERS R2-R9
        GSYSSRV SETRWM,K,1,-              ;SET RESOURCE WAIT MODE
                <R4>                      ;REGISTER R4
        GSYSSRV SETSFM,K,1,-              ;SET SYSTEM SERVICE FAILURE MODE
                <R4>,EXC_MASK             ;REGISTER R4, AND EXECPTION MASK
        GSYSSRV SETSWM,K,1,-              ;SET PROCESS SWAP MODE
                <R4>                      ;REGISTER R4
        GSYSSRV SUSPND,K,2,-              ;SUSPEND PROCESS
                <R2,R3,R4,R5>             ;REGISTERS R2-R5
        GSYSSRV TRNLOG,ALL,6,-            ;TRANSLATE LOGICAL NAME
                <R2,R3,R4,R5,R6,R7,R8>    ;REGISTERS R2-R8
        GSYSSRV ULKPAG,K,3,-              ;UNLOCK PAGE FROM MEMORY
                <R2,R3,R4,R5,R6,R7,R8>    ;REGISTERS R2-R8
        GSYSSRV ULWSET,K,3,-              ;UNLOCK PAGES FROM WORKING SET
                <R2,R3,R4,R5,R6,R7,R8>    ;REGISTERS R2-R8
        GSYSSRV UNWIND,ALL,2,-            ;UNWIND PROCEDURE CALL STACK
                <R2,R3,R4,R5>             ;REGISTERS R2-R5
        GSYSSRV WAITFR,K,1,-              ;WAIT FOR EVENT FLAG
                <R2,R3,R4,R5,R6>          ;REGISTERS R2-R6.  IF R8 IS EVER USED
                                          ;THE RMS SYCHRONIZATION CODE MUST BE
                                          ;MODIFIED TO SAVE IT ALSO.
        GSYSSRV WAKE,K,2,-                ;WAKE PROCESS
                <R2,R3,R4,R5>             ;REGISTERS R2-R5
        GSYSSRV WFLAND,K,2,-              ;WAIT FOR LOGICAL AND OF EVENT FLAGS
                <R2,R3,R4,R5,R6>          ;REGISTERS R2-R6
        GSYSSRV WFLOR,K,2,-               ;WAIT FOR LOGICAL OR OF EVENT FLAGS
                <R2,R3,R4,R5,R6>          ;REGISTERS R2-R5
        GSYSSRV BRDCST,ALL,2,-            ;BROADCAST TO TERMINALS
                <R2,R3,R4,R5,R6>          ;REGISTERS R2-R6
        GSYSSRV DCLCMH,K,3,-              ;DECLARE CHANGE MODE HANDLER
                <R4>                      ;SAVE R4
        GSYSSRV SETPFM,K,4,-              ;SET PAGE FAULT MONITORING
                <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;REGISTERS R2-R11
        GSYSSRV GETMSG,ALL,5,-            ;GET MESSAGE
                <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;REGISTERS R2-R11
        GSYSSRV DERLMB,K,1,-              ;DECLARE ERROR LOG MAILBOX
                <R2,R3,R4,R5>             ;REGISTERS R2-R5
        GSYSSRV CANEXH,K,1,-              ;CANCEL EXIT HANDLER
                <R2,R3,R4,R5>             ;REGISTERS R2-R5
        GSYSSRV GETCHN,K,5,-              ;GET CHANNEL INFORMATION
```

```
                <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;REGISTERS R2-R11
        GSYSSRV GETDEV,K,5,-            ;GET DEVICE INFORMATION
                <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;REGISTERS R2-R11
        GSYSSRV GETJPI,K,7,-            ;GET JOB PROCESS INFORMATION
                <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;REGISTERS R2-R11
        GSYSSRV PUTMSG,ALL,3,-          ;PUT FORMATTED ERROR MESSAGE
                <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;REGISTERS R2-R11
        GSYSSRV EXCMSG,ALL,2,-          ;OUTPUT EXCEPTION SUMMARY MESSAGE
                <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;REGISTERS R2-R11
        GSYSSRV SNDACC,E,2,-            ;SEND MSG TO ACOUNTING MANAGER
                <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;REGISTERS R2-R11
        GSYSSRV SETIME,K,1,-            ;SET SYSTEM TIME
                <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;REGISTERS R2-R11
        GSYSSRV SETPRV,K,4,-            ;SET PRIVILEGES
                <R2,R3,R4,R5,R6,R7,R8>  ;REGISTERS R2-R8
                .PAGE
;
;       SPECIAL VECTORS FOR AST DELIVERY AND CLEARING
;
;       SYS$CLRAST CLEARS THE CURRENTLY ACTIVE AST STATUS
;
;       SYS$GL_ASTRET CONTAINS THE VALUE OF THE RETURN ADDRESS FROM
;       THE CALL INSTRUCTION USED TO DISPATCH AN AST. THIS VALUE CAN
;       BE USED WHEN SEARCHING UP THE STACK FOR THE AST CALL FRAME.
;
                .IF     NDF,MPSWITCH
                .IF     NDF,RMSSWITCH
                .IF     DF,LIBSWITCH
                .PSECT  $$$0000,QUAD
                .IFF    ;LIBSWITCH
                .PSECT  $$$000,QUAD
                .ENDC   ;LIBSWITCH
                .ALIGN  QUAD
                .IF     DF,LIBSWITCH
SYS$CLRAST::                            ;CLEAR ACTIVE AST
                .BLKL   2               ;
                .IFF    ;LIBSWITCH
                .WORD   ^M<>            ;SAVE NO REGISTERS
                CHMK    #CLRAST         ;DO SPECIAL CHMK
                RET                     ;AND RETURN
CLRAST=0                                ;
                .ENDC   ;LIBSWITCH
                .ALIGN  QUAD            ;
                .IF     DF,LIBSWITCH
SYS$GL_ASTRET::                         ;
                .BLKL   1
SYS$GL_COMMON::                         ;ADDRESS OF CORE COMMON DESCRIPTOR
                .BLKL   1               ;
                .IFF    ;LIBSWITCH
                .LONG   EXE$ASTRET      ;RETURN ADDRESS FROM AST DISPATCHING
                                        ;CALL
                .LONG   CTL$GQ_COMMON   ;ADDRESS OF ''CORE COMMON'' DESCRIPTOR
                .ENDC   ;LIBSWITCH

;
; ENTRY VECTOR FOR CONDITION HANDLER SEARCH. LIB$SIGNAL USES THIS VECTOR
```

```
;  TO SHARE EXCEPTION'S CODE TO SEARCH FOR AND CALL CONDITION HANDLERS.
;  THIS ENTRY IS NOT CALLED; RATHER, IT IS JUMPED TO. NO RETURN IS MADE.
;
        .ALIGN  QUAD
        .IF DF  LIBSWITCH
SYS$SRCHANDLER::
        .IFF    ;LIBSWITCH
        JMP     G^EXE$SRCHANDLER        ;JUMP TO COMMON CODE
        .IFT    ;LIBSWITCH
        .BLKQ   1                       ;RESERVE SPACE
        .ENDC   ;LIBSWITCH

        .ENDC   ;RMSSWITCH
;
;  NOTE THAT THE CODE IN PSECT $$$000 AT THIS POINT CANNOT EXCEED 320 (HEX)
;  WITHOUT MODIFYING THE RMS SYNCHRONIZATION CODE WHICH PRECEDES THE RMS
;  VECTORS WHICH CANNOT BE MOVED.
;
.PAGE
;
; Set up the base for the RMS service codes. We leave a hole so that
; other exec mode system services can be defined later in this module.
; The hole is defined by the offset between ECASCTR and RCASCTR; it
; is checked with an ASSUME at the end of all service definitions.
;
        .IF     NDF,LIBSWITCH
RCASCTR=ECASCTR+10
        .ENDC

        .IF     DF,RMSSWITCH
;
;  CASE DISPATCHER FOR RMS SERVICES
;
;       R0 HAS SERVICE DISPATCH CODE.
;       IF IN RANGE DISPATCHES TO APPROPRIATE RMS SERVICE,
;       ELSE SIMPLY DOES AN RSB
;
        .PSECT  $$$RMSVEC,BYTE,NOWRT    ;MUST BE FIRST PSECT IN RMS
RMS$DISPATCH:                           ;MUST BE FIRST CODE IN FIRST RMS PSECT
        CASEW   R0,S^#RCASMIN,S^#RCASMAX

RCASE:
        .IFTF   ;RMSSWITCH
        .IF     NDF,LIBSWITCH
RCASMIN=RCASCTR
        .ENDC
        .IFF    ;RMSSWITCH
        .PAGE
;++
;
;  RMS SERVICES
;
;
;  RMS SYNCHRONIZATION ROUTINE
```

```
;
;  THE FOLLOWING ROUTINE IS USED BY THE VARIOUS RMS SERVICES IN ORDER
;  TO AWAIT I/O COMPLETION.  THE ROUTINE IS IN THE VECTOR AREA IN ORDER
;  TO WAIT AT THE CALLER'S MODE,  THUS ALLOWING AST ACTIVITY FOR EITHER
;  USER OR SUPERVISOR MODE, OR BOTH.
;
;  THE FAB/RAB IS CHECKED FOR A LEGAL BLOCK ID, I.E., A 1 OR 3, AND
;  AN ERROR RETURNED IF INVALID.  THE STRUCTURE IS NOT REPROBED.
;
;  NOTE THAT EACH RMS SERVICE VECTOR TERMINATES WITH A BRANCH TO THIS
;  ROUTINE.
;
;  THIS ROUTINE ASSUMES THAT THE FOLLOWING REGISTERS HAVE BEEN SET BY THE
;  EXITING RMS EXEC-LEVEL CODE WHENEVER A STALL IS REQUIRED:
;
;       R3      EFN TO WAIT ON
;       R8      RAB/FAB ADDRESS TO WAIT ON
;       R4      (RMSWAIT_BR ENTRY POINT ONLY, $WAIT SERVICE) FLAG FOR WAIT TYPE
;               (0 = SAME RAB, 1 = DIFFERENT RABS)
;
;--
        .IF NDF,LIBSWITCH
        .PSECT  $$$000,QUAD
        .IFF    ;LIBSWITCH
        .PSECT  $$$0000,QUAD
        .IFTF   ;LIBSWITCH
        .BLKB   ^X320-<.-VECBASE>
        .IFT    ;LIBSWITCH
RMSWAIT_IO_DONE:
;
;  SET A FLAG IN THE USER'S CONTROL BLOCK THAT TELLS RMS THAT THE PROCESS
;  IS WAITING ON THIS FAB/RAB.  WHEN RMS IS INITIALIZING FOR A NEW OPERATION
;  IT CHECKS THIS FLAG AND REJECTS THE NEW OPERATION IF THE CONTROL BLOCK
;  IS WAITING ON A PREVIOUS OPERATION.  THIS PREVENTS A HANG CONDITION
;  CAUSED BY USING THE SAME STS/STV FIELD FOR 2 OPERATIONS AT ONCE.
;  FAB$B_BLN = RAB$B_BLN
;
        BISB    #1,RAB$B_BLN(RB)        ;LOW BIT OF BLN FIELD IS THE FLAG

;
;  THE ARGUMENTS ARE PUSHED ON THE STACK AND THE AP SET UP AS IF A 'CALLS'
;  INSTRUCTION WERE BEING EXECUTED.  THE CHANGE MODE TO KERNEL SERVICE IS
;  EXECUTED DIRECTLY.  THIS SAVES THE OVERHEAD OF A 'CALLS' INSTRUCTION.
;  RB MUST NOT BE DESTROYED BY ANY OF THE SERVICES USED HERE.
;
        PUSHL   R3                      ;EVENT FLAG TO WAIT FOR
        MOVAB   -4(SP),AP               ;SET UP AP AS IF USING CALLS INSTR.
        PUSHL   #1                      ;NUMBER OF ARGUMENTS
USERWAIT:
        CHMK    I^#WAITFR               ;DO 'NAKED' WAITFR TO SAVE CALLS TIME
;
;  CHECK TO SEE IF THE USER STRUCTURE POINTED TO BY RB IS STILL VALID BY
;  CHECKING THE BLOCK ID TO BE SURE THAT IT IS EITHER A RAB (BID=1) OR
;  A FAB (BID=3).  THIS WON'T CATCH THE CASE WHERE WHAT SHOULD HAVE BEEN
;  A FAB NOW LOOKS LIKE A RAB OR VICE VERSA  BUT WILL CATCH EVERYTHING
;  ELSE.  IF THE STRUCTURE IS NOT READABLE OR WRITEABLE THEN THE USER
```

```
;  WILL GET AN ACCESS VIOLATION.  THE BID FOR A FAB/RAB IS AT BYTE 0,
;  THE STS FOR A FAB/RAB IS AT BYTE 8.
;
10$:    BLBC    (R8),30$                ;NOT SET, THEN NOT A FAB OR RAB
        BITB    #^B11111100,(R8)        ;IS IT A 1 OR 3?
        BNEQ    30$                     ;NEQ NO SO BLOW THE WHISTLE
        MOVL    8(R8),R0                ;GET RMS STATUS CODE
        BEQL    20$                     ;AND WAIT AGAIN IF NOT SET
        BICB    #1,RAB$B_BLN(R8)        ;CLEAR WAITING FLAG
        BLBC    R0,30$                  ;BRANCH IF FAILURE CODE
        RET                             ;RETURN TO CALLER
;
;  CLEAR THE RMS EVENT FLAG, CHECK STATUS AGAIN AND WAIT 1 MORE TIME IF
;  OPERATION STILL NOT DONE.  THE APPROPRIATE ARGUMENTS FOR THE CLREF
;  AND SETEF (IF EXECUTED) REMAIN ON THE STACK FROM THE WAITFR ABOVE.
;  THE AP MUST BE PRESERVED.
;
20$:    CHMK    I^#CLREF                ;DO A 'NAKED' CLREF, THE ARGUMENTS
                                        ;ARE ON STACK AND AP STILL SET UP
                                        ;FROM THE WAITFR ABOVE
        TSTL    8(R8)                   ;AND RE-CHECK STATUS
        BEQL    USERWAIT                ;BRANCH TO WAIT FOR FLAG AGAIN..
                                        ;... IF STATUS STILL ZERO
        CHMK    I^#SETEF                ;I/O COMPLETE - LEAVE EFN SET
        BRB     10$                     ;AND RESTORE R0 STATUS CODE
;
;  BRANCH TO CHECK STATUS CODE FOR ERROR OR SEVERE ERROR
;  A SUCCESS STATUS IN R0 (FROM THE $WAITFR) INDICATES AN INVALID FAB/RAB.
;
30$:    BRW     RMS_ERR_BR
;
;  ENTRY HERE FROM $WAIT SERVICE.  THIS SERVES AS AN EXTENDED BRANCH
;  TO THE $WAIT SYNCHRONIZATION CODE IN THE Y$CMODE PSECT.
;
RMSWAIT_BR:
        JSB     @#RMS_WAIT_SYNC                 ;DO $WAIT SYNCHRONIZATION
;
;
;  ENTRY HERE FROM EACH VECTOR
;  CHECK FOR POSSIBLE STALL
;
RMSCHK_STALL:
        CMPW    R0,#RMS$_STALL&^XFFFF   ;IS THE STATUS CODE I/O STALL?
        BEQL    RMSWAIT_IO_DONE         ;BRANCH IF YES
        RET                             ;BACK TO CALLER
        .ALIGN  QUAD
        .IIF    ;LIBSWITCH
        .BLKB   ^X48                    ;THIS TAKES THE SPACE OF THE CODE
                                        ;WHEN GENERATING THE GLOBAL SYMBOLS
        .ENDC   ;LIBSWITCH
        .IFF    ;RMSSWITCH
.PAGE
;
;
;  DEFINE RMS SERVICES
;
```

```
        .IF     NDF,LIBSWITCH
RMSSYNC=RMSCHK_STALL
        .ENDC
        .ENDC   ;RMSSWITCH
;
;  HIGH USE RECORD OPERATIONS
;
        RMSSRV  DELETE          ;DELETE A RECORD
        .NLIST CND
        RMSSRV  FIND            ;FIND RECORD
        RMSSRV  FREE            ;RELEASE LOCK ON ALL RECORDS
        RMSSRV  GET             ;GET A RECORD
        RMSSRV  PUT             ;PUT A RECORD
        RMSSRV  READ            ;READ A BLOCK
        RMSSRV  RELEASE         ;RELEASE LOCK ON NAMED RECORD
        RMSSRV  UPDATE          ;REWRITE EXISTING RECORD
        .IF     NDF,RMSSWITCH
        .IF     NDF,LIBSWITCH
RMSSYNC=RMSWAIT_BR              ;REDEFINE FOR $WAIT ONLY
        .ENDC
        .ENDC   ;RMSSWITCH
        RMSSRV  WAIT            ;STALL FOR RECORD OPERATION COMPLETE
        .IF     NDF,RMSSWITCH
        .IF     NDF,LIBSWITCH
RMSSYNC=RMSCHK_STALL            ;RESTORE STANDARD SYNC ADDR
        .ENDC
        .ENDC   ;RMSSWITCH
        RMSSRV  WRITE           ;WRITE BLOCK
;
;  LOWER USAGE OPERATIONS
;
      RMSSRV  CLOSE             ;CLOSE FILE
      RMSSRV  CONNECT           ;CONNECT RAB
      RMSSRV  CREATE            ;CREATE FILE
      RMSSRV  DISCONNECT        ;DISCONNECT RAB
      RMSSRV  DISPLAY           ;DISPLAY FILE INFORMATION
      RMSSRV  ERASE             ;ERASE (DELETE) FILE
      RMSSRV  EXTEND            ;EXTEND FILE ALLOCATION
      RMSSRV  FLUSH             ;FINISH I/O ACTIVITY FOR STREAM
      RMSSRV  MODIFY            ;MODIFY FILE ATTRIBUTES
      RMSSRV  NXTVOL            ;NEXT VOLUME
      RMSSRV  OPEN              ;OPEN FILE
      RMSSRV  REWIND            ;REWIND FILE
      RMSSRV  SPACE             ;POSITION FOR TRANSFER
      RMSSRV  TRUNCATE          ;TRUNCATE FILE
      RMSSRV  ENTER             ;ENTER FILENAME INTO DIRECTORY
      RMSSRV  PARSE             ;PARSE FILENAME SPECIFICATION
      RMSSRV  REMOVE            ;REMOVE FILENAME FROM DIRECTORY
      RMSSRV  RENAME,NARG=4     ;RENAME A FILE
      RMSSRV  SEARCH            ;SEARCH A FILE DIRECTORY
      RMSSRV  SETDDIR,NARG=3,NOSYNC=1
                                ;SET DEFAULT DIRECTORY STRING
      RMSSRV  SETDFPROT,REGS=<R2,R3>,NARG=2,NOSYNC=1
                                ;SET DEFAULT FILE PROTECTION MASK
      RMSSRV  SSVEXC,REGS=<>,NOSYNC=1
                                ;GENERATE SYS SERV EXCEPTION
```

```
        RMSSRV  RMSRUNDWN,NARG=2,NOSYNC=1
                                ;PERFORM RUNDOWN ON RMS FILES
        RMSSRV  RMSRUHNDLR,NARG=5,NOSYNC=1
                                ;RMS Recovery Unit Handler
        RMSSRV  FILESCAN,NARG=3,NOSYNC=1
                                ;Perform syntax check for file specs
;
; ADD NEW RMS SERVICES IN FRONT OF THIS CODE!
;
; Now we add special non-vector code.  Because of the CASE instruction
; used at the front of RMS, this code (and any future additional code)
; must be the last element of the RMS area.
;
        GCOMPSRVB               ;Helper branch to error processing
        .IF     NDF,MPSWITCH
        .IF     NDF,RMSSWITCH
        .IF     NDF,LIBSWITCH
RMS_ERR_BR:
        JMP     @#RMS_ERR
        .ENDC   ;LIBSWITCH
        .ENDC   ;RMSSWITCH
        .ENDC   ;MPSWITCH
        GCOMPSRVE       1

        .IF     NDF,RMSSWITCH

; NOTE:  RMSVECEND MARKS THE END OF THE CURRENTLY DEFINED RMS VECTORS.
;        SSVECREG2 MARKS THE START OF THE SECOND REGION OF SYSTEM
;        SERVICE VECTORS.  THERE IS EMPTY SPACE BETWEEN THESE REGIONS
;        FOR FUTURE RMS VECTORS.  IF NECESSARY, THIS SPACE CAN ALSO
;        BE USED FOR SYSTEM SERVICE VECTORS BY BACKING UP SSVECREG2
;        (TOWARDS THE RMS VECTORS) AND ADDING NEW SYSTEM SERVICE VECTORS
;        BEFORE THE ALREADY DEFINED ONES.  IN OTHER WORDS, THESE TWO
;        VECTOR REGIONS MAY GROW TOWARDS EACH OTHER.  IF THEY COLLIDE,
;        AN ASSEMBLY ERROR IS GENERATED.

        .IF     DF,LIBSWITCH
        .PSECT  $$$0000,QUAD
        .IFF    ;LIBSWITCH
        .PSECT  $$$000,QUAD             ; CMODSSDSP
        .ENDC   ;LIBSWITCH

RMSVECEND:
.=VECBASE+^X5C0
SSVECREG2:                      ; START OF SYSTEM SERVICE VECTOR REGION 2
        .IF     GT,RMSVECEND-SSVECREG2
        .ERROR                  ; RMS VECTORS EXCEEDED PREALLOCATED SPACE ;
        .ENDC
        .ENDC   ;RMSSWITCH
        .ENDC   ;MPSWITCH

        .PAGE
        .SBTTL  REGION 2 OF SYS. SERV. VECTOR DEFINITIONS

;
```

```
; Note: Service codes for exec mode services in this region are
; reserved by the offset defined above between RCASCTR and ECASCTR.
; If the ASSUME at the end of this section breaks, the offset must
; be increased.
;
        GSYSSRV ENQ,K,11,-              ; ENQUEUE
                <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; REGISTERS R2-R11
        GSYSSRV DEQ,K,4,-               ; DEQUEUE
                <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; REGISTERS R2-R11
        GCOMPSRVB ENQW,-                ; ENQUEUE AND WAIT
                <ENQ_MASK ! WAITFR_MASK ! CLREF_MASK ! SETEF_MASK>
        .IF     NDF,MPSWITCH
        .IF     NDF,RMSSWITCH
        .IF     NDF,LIBSWITCH
        CHMK    #ENQ                    ; EXECUTE ENQ SYSTEM SERVICE
        CMPW    R0,#SS$_SYNCH           ; IF COMPLETED SYNCHRONOUSLY
        BNEQ    10$
5$:     RET                             ; THEN RETURN WITHOUT ANY WAITING
10$:    BLBC    R0,5$                   ; DON'T WAIT IF ERROR
        PUSHL   ENQ$_LKSB(AP)           ; OTHERWISE GET IOSB ADDRESS IF SPECIFIED
        BRB     QIO_ENQ_SYNCH           ; AND USE COMMON SYNCH CODE
        .ENDC   ;LIBSWITCH
        .ENDC   ;RMSSWITCH
        .ENDC   ;MPSWITCH
        GCOMPSRVE       3               ; RESERVE 3 QUADWORDS FOR VECTOR
        GSYSSRV SETSSF,K,1,-            ; SET SYSTEM SERVICE FILTER MASK
                <R4>                    ; REGISTER R4
        GSYSSRV SETSTK,K,3,-            ; SET STACK LIMITS
                <R2,R3,R4>              ; REGISTERS R2,R3,R4
        GSYSSRV GETSYI,K,7,-            ; GET SYSTEM INFORMATION
                <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; REGISTERS R2-R11
        GSYSSRV IMGFIX,ALL,0,-          ; IMAGE ADDRESS RELOCATION FIXUP
                <R2,R3,R4,R5>           ; REGISTERS R2-R5
        GCOMPSRVB       IMGFIX_2,-      ; ********** TEMP **********
                <0>
        GCOMPSRVE       1               ; ********** TEMP **********
        GSYSSRV GETDVI,K,8,-            ; GET DEVICE AND VOLUME INFORMATION
                <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; REGISTERS R2-R11
        GCOMPSRVB GETDVIW,-             ; GET DEVICE INFORMATION AND WAIT
                <GETDVI_MASK ! GETJPI_SYNCH_MASK>
        .IF     NDF,MPSWITCH
        .IF     NDF,RMSSWITCH
        .IF     NDF,LIBSWITCH
        CHMK    I^#GETDVI
        BRB     GETJPI_COMMON
        .ENDC   ;LIBSWITCH
        .ENDC   ;RMSSWITCH
        .ENDC   ;MPSWITCH
        GCOMPSRVE       1
        GCOMPSRVB GETJPIW,-             ; GET JOB/PROCESS INFORMATION AND WAIT
                <GETJPI_MASK ! GETJPI_SYNCH_MASK>
        .IF     NDF,MPSWITCH
        .IF     NDF,RMSSWITCH
        .IF     NDF,LIBSWITCH
        CHMK    I^#GETJPI
```

```
GETJPI_COMMON:
        JMP     @#GETJPI_SYNCH
        .ENDC   ;LIBSWITCH
        .ENDC   ;RMSSWITCH
        .ENDC   ;MPSWITCH
        GCOMPSRVE       2
        GCOMPSRVB GETSYIW,-             ; GET SYSTEM INFORMATION AND WAIT
                <GETSYI_MASK ! GETJPI_SYNCH_MASK>
        .IF     NDF,MPSWITCH
        .IF     NDF,RMSSWITCH
        .IF     NDF,LIBSWITCH
        CHMK    I^#GETSYI
        BRB     GETJPI_COMMON
        .ENDC   ;LIBSWITCH
        .ENDC   ;RMSSWITCH
        .ENDC   ;MPSWITCH
        GCOMPSRVE       1
        GCOMPSRVB SNDJBCW,-             ; SEND TO JOB CONTROLLER AND WAIT
                <SNDJBC_MASK ! GETJPI_SYNCH_MASK>
        .IF     NDF,MPSWITCH
        .IF     NDF,RMSSWITCH
        .IF     NDF,LIBSWITCH
        CHME    I^#SNDJBC               ; SEND TO JOB CONTROLLER
        BRB     GETJPI_COMMON
        .ENDC   ;LIBSWITCH
        .ENDC   ;RMSSWITCH
        .ENDC   ;MPSWITCH
        GCOMPSRVE       1
        GCOMPSRVB SYNCH,-               ; SYNCHRONIZE EFN AND IOSB
                <WAITFR_MASK ! CLREF_MASK ! SETEF_MASK>
        .IF     NDF,MPSWITCH
        .IF     NDF,RMSSWITCH
        .IF     NDF,LIBSWITCH
        PUSHL   SYNCH$_IOSB(AP)         ; GET ADDRESS OF IOSB IF SPECIFIED
;
; CONDITION CODES SET FROM PUSH OF IOSB ADR ONTO STACK
; THE EFN STATE AND IOSB STATUS MAY HAVE ONLY THE FOLLOWING COMBINATIONS
;       EFN CLEAR, (IOSB) = 0
;       EFN SET, (IOSB) NON ZERO
;       EFN SET, (IOSB) CLEAR - the EFN was set by another I/O operation
;
; IF THE EFN COULD BE CLEAR AND (IOSB) WAS NON-ZERO, THIS SERVICE WOULD
; EXIT WITH THE EVENT FLAG CLEAR WHICH IS NOT CORRECT.
;
QIO_ENQ_SYNCH:
        BEQL    50$                     ; BRANCH IF NO IOSB SPECIFIED
        TSTW    @(SP)                   ; IS COMPLETION STATUS SET?
        BNEQ    40$                     ; BRANCH IF SET
10$:    CHMK    I^#WAITFR               ; MUST WAIT FOR EFN TO BE SET
        TSTW    @(SP)                   ; COMPLETION STATUS SET YET?
        BEQL    30$                     ; BRANCH IF NOT
20$:    RET                             ; YES, RETURN STATUS
30$:    BLBC    R0,20$                  ; IF ERROR, RETURN STATUS
        CHMK    I^#CLREF                ; NO, CLEAR EVENT FLAG
        TSTW    @(SP)                   ; AND IF STILL NOT DONE
```

```
        BEQL    10$                     ; WAIT SOME MORE
        CHMK    I^#SETEF                ; OTHERWISE EXIT WITH IT SET
40$:    MOVL    S^#SS$_NORMAL,R0        ; FORCE NORMAL SUCCESS
        RET                             ; AND RETURN
;
; NO IOSB GIVEN, JUST WAIT FOR THE EVENT FLAG TO BE SET
;
50$:    CHMK    I^#WAITFR               ; WAIT FOR SPECIFIED EVENT FLAG
        RET                             ; AND RETURN
        .ENDC   ;LIBSWITCH
        .ENDC   ;RMSSWITCH
        .ENDC   ;MPSWITCH
        GCOMPSRVE       6               ; RESERVE 6 QUADWORDS FOR VECTOR
        GSYSSRV ERAPAT,K,3,-            ; GENERATE A SECURITY  ERASE PATTERN
                <R4>                    ; SAVE R4
        GSYSSRV CRELNT,K,8,-            ; CREATE LOGICAL NAME TABLE
                <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;REGISTERS R2-R11
        GSYSSRV CRELNM,K,5,-            ; CREATE LOGICAL NAME
                <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;REGISTERS R2-R11
        GSYSSRV DELLNM,K,3,-            ; DELETE LOGICAL NAME
                <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;REGISTERS R2-R11
        GSYSSRV TRNLNM,K,5,-            ; TRANSLATE LOGICAL NAME
                <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;REGISTERS R2-R11
        GSYSSRV GETLKI,K,7,-            ; GET LOCK INFORMATION
                <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;REGISTERS R2-R11
        GCOMPSRVB GETLKIW,-             ; GET LOCK INFORMATION AND WAIT
                <GETLKI_MASK ! WAITFR_MASK ! CLREF_MASK ! SETEF_MASK>
        .IF     NDF,MPSWITCH
        .IF     NDF,RMSSWITCH
        .IF     NDF,LIBSWITCH
        CHMK    I^#GETLKI
        BLBC    R0,10$                  ; DON'T WAIT IF ERROR
        PUSHL   GETLKI$_IOSB(AP)        ; OTHERWISE GET IOSB ADDRESS IF SPECIFIED
        BRB     QIO_ENQ_SYNCH           ; AND USE COMMON SYNCH CODE
10$:    RET                             ; RETURN ON ERROR
        .ENDC   ;LIBSWITCH
        .ENDC   ;RMSSWITCH
        .ENDC   ;MPSWITCH
        GCOMPSRVE       2               ; RESERVE 2 QUADWORDS FOR VECTOR

        GSYSSRV ASCTOID,E,3,-           ;ASCII TO IDENTIFIER CONVERSION
                <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;REGISTERS R2-R11
        GSYSSRV FINISH_RDB,E,1,-        ;FINISH RDB CONTEXT STREAM
                <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;REGISTERS R2-R11
        GSYSSRV IDTOASC,E,6,-           ;IDENTIFIER TO ASCII CONVERSION
                <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;REGISTERS R2-R11
        GSYSSRV BRKTHRU,K,11,-          ;BREAK THROUGH WRITES
                <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;REGISTERS R2-R11
        GSYSSRV GRANTID,ALL,5,-         ;GRANT IDENTIFIER TO PROCESS
                <R2,R3>                 ;REGISTERS R2-R3
        GSYSSRV REVOKID,ALL,5,-         ;REVOKE IDENTIFIER FROM PROCESS
                <R2,R3>                 ;REGISTERS R2-R3
        GSYSSRV CHKPRO,K,1,-            ;GENERAL PROTECTION CHECK ROUTINE
                <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;REGISTERS R2-R11
        GCOMPSRVB BRKTHRUW,-            ; BREAK THOUGH WRITE AND WAIT
                <BRKTHRU_MASK ! GETJPI_SYNCH_MASK>
```

```
        .IF     NDF,MPSWITCH
        .IF     NDF,RMSSWITCH
        .IF     NDF,LIBSWITCH
        CHMK    I^#BRKTHRU
        BRW     GETJPI_COMMON
        .ENDC   ;LIBSWITCH
        .ENDC   ;RMSSWITCH
        .ENDC   ;MPSWITCH
        GCOMPSRVE       2
        GSYSSRV GETQUI,E,7,-                    ;GET QUEUE INFORMATION
                <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;REGISTERS R2-R11
        GCOMPSRVB GETQUIW,-                     ;GET QUEUE INFORMATION AND WAIT
                <GETQUI_MASK ! GETJPI_SYNCH_MASK>
        .IF     NDF,MPSWITCH
        .IF     NDF,RMSSWITCH
        .IF     NDF,LIBSWITCH
        CHME    I^#GETQUI
        BRW     GETJPI_COMMON
        .ENDC   ;LIBSWITCH
        .ENDC   ;RMSSWITCH
        .ENDC   ;MPSWITCH
        GCOMPSRVE       2

;
        CJF$KCASCTR = 16424
;
        LDBSRV  CJF$, ALLJDR,       K,   <R4>
        LDBSRV  CJF$, ASSJNL,       K,   <R4>
        LDBSRV  CJF$, CONUIC,       K,   <R4>
        LDBSRV  CJF$, CREJNL,       K,   <R4>
        LDBSRV  CJF$, DEALJDR,      K,   <R4>
        LDBSRV  CJF$, DEASJNL,      ALL, <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
        LDBSRV  CJF$, DEASJNL_INT,  K,   <R4>
        LDBSRV  CJF$, DELJNL,       K,   <R4>
        LDBSRV  CJF$, DMTJMD,       K,   <R4>
        LDBSRV  CJF$, DSPJNL,       K,   <R4>
        LDBSRV  CJF$, GETJNL,       K,   <R4>
        LDBSRV  CJF$, GETRUI,       K,   <R4>
        LDBSRV  CJF$, MODFLT,       K,   <R4>
        LDBSRV  CJF$, POSJNL,       K,   <R4>
        LDBSRV  CJF$, READJNL,      K,   <R4>
        LDBSRV  CJF$, RECOVER,      K,   <R4>
        LDBSRV  CJF$, MNTJMD,       K,   <R4>
        LDBSRV  CJF$, CRENWV,       K,   <R4>
        LDBSRV  CJF$, CONJNLF,      K,   <R4>
        LDBSRV  CJF$, DCNJNLF,      K,   <R4>
        LDBSRV  CJF$, FORCEJNL,     ALL, <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
        LDBSRV  CJF$, FORCEJNLW,    ALL, <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
        LDBSRV  CJF$, WRITEJNL,     ALL, <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
        LDBSRV  CJF$, WRITEJNLW,    ALL, <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
        LDBSRV  CJF$, GETCJI,       K,   <R4>
        LDBSRV  CJF$, DMTJMDW,      K,   <R4>, 4, 5, DMTJMD
        LDBSRV  CJF$, MODFLTW,      K,   <R4>, 4, 5, MODFLT
        LDBSRV  CJF$, POSJNLW,      K,   <R4>, 4, 5, POSJNL
        LDBSRV  CJF$, READJNLW,     K,   <R4>, 4, 5, READJNL
        LDBSRV  CJF$, RECOVERW,     K,   <R4>, 5, 6, RECOVER
```

```
;
        RUF$KCASCTR = 16400
;
        LDBSRV  RUF$,  REENTERRU,     K,  <R2,R3,R4,R5,R6>
        LDBSRV  RUF$,  STARTRU,       K,  <R2,R3,R4,R5,R6>
        LDBSRV  RUF$,  PHASE1,        K,  <R2,R3,R4,R5,R6>
        LDBSRV  RUF$,  PHASE2,        K,  <R2,R3,R4,R5,R6>
        LDBSRV  RUF$,  CANCELRU,      K,  <R2,R3,R4,R5,R6>
        LDBSRV  RUF$,  MARKPOINTRU,   K,  <R2,R3,R4,R5,R6>
        LDBSRV  RUF$,  RESETRU,       K,  <R2,R3,R4,R5,R6>
        LDBSRV  RUF$,  DCLRUH,        K,  <R2,R3,R4,R5,R6>
        LDBSRV  RUF$,  CANRUH,        K,  <R2,R3,R4,R5,R6>
        LDBSRV  RUF$,  RUSTATUS,      K,  <R2,R3,R4,R5,R6>
;
; End Recovery Unit consists of a two-phase commit, so we call each
; phase separately.
;
        GCOMPSRVB ENDRU, <PHASE1_MASK ! PHASE2_MASK>, RUF$ ; End Recovery Unit
        .IF     NDF,MPSWITCH
        .IF     NDF,RMSSWITCH
        .IF     NDF,LIBSWITCH
        CHMK    I^#PHASE1
        BLBC    R0,10$
        CHMK    I^#PHASE2
10$:    RET
        .ENDC   ;LIBSWITCH
        .ENDC   ;RMSSWITCH
        .ENDC   ;MPSWITCH
        GCOMPSRVE       2
        GSYSSRV MTACCESS,K,6,-          ;Mag tape installation specific access routine
                <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;REGISTERS R2-R11
;
; End of system service vector definitions. New system services are
; to be added at this point.
;
        .IF     NDF,MPSWITCH
        .IF     NDF,LIBSWITCH
        ASSUME  RCASMIN GE ECASCTR      ;Exec service codes must not collide with RMS
        .ENDC   ;LIBSWITCH
        .ENDC   ;MPSWITCH

        .PAGE
        .IF     NDF,RMSSWITCH
        .IF     NDF,LIBSWITCH           ;GENERATE CODE IF NOT LIBRARY FORM
        .IF     NDF,MPSWITCH
        .PSECT  $$$000,BYTE
CLIJMP:
        PUSHL   @#CTL$AL_CLICALBK       ;PIC JUMP FOR CLI CALLBACK
        JMP     @(SP)+
        .BLKB   <SGN$C_SYSVECPGS@9>-<.-VECBASE> ;FILL REMAINDER OF RESERVED PAGES

        .PAGE
        .SBTTL  ILLEGAL CHME OR CHMK CODE VALUE HANDLING
;
```

```
;
;       END OF CHME DISPATCH TABLE
;

        .PSECT  Y$CMODE,QUAD
        JSB     @CTL$GL_RMSBASE         ;SEE IF RMS DOES THIS SERVICE
                                        ;  (R0 HAS CHME CODE)
        JSB     EXE$LOAD_EDISP          ; CALL LOADABLE CODE DISPATCHERS

        TSTB    @#CTL$GB_SSFILTER       ; ANY INHIBIT BITS ON?
        BEQL    5$                      ; NO, ALL OKAY
        MOVZWL  #SS$_INHCHME,R1         ; YES, SET THE EXCEPTION CODE
        BRW     INHEXCP1                ; DEAL WITH BAD CODE

5$:     MOVL    @#CTL$GL_USRCHME,R1     ; GET PER-PROCESS USER CHME VECTOR
        BEQL    10$                     ; NOT PRESENT, TRY SYSTEM WIDE

;
;       CALL PER-PROCESS "USER" SUPPLIED PLUG-ON HANDLER FOR CHME
;       WITH UNRECOGNIZED CODES.
;
;       R0 - CODE FROM CHME/CHMK (LONGWORD)
;       R1 - ADDRESS OF ROUTINE
;       (SP) - RETURN ADDRESS IN CASE CODE IS NOT LEGAL.
;              IF AN RSB IS ISSUED, THEN THE SYSTEM-WIDE HANDLER WILL BE
;              GIVEN AN OPPORTUNITY BEFORE DECIDING THAT THE CODE IS REALLY ILLEGAL.
;               (NORMAL RETURN IS A RET AFTER PERFORMING FUNCTION)
;
        JSB     (R1)                    ; CALL PER-PROCESS USR CHME HANDLER
                                        ; RETURNS ONLY IF ILLEGAL CODE
10$:    MOVL    L^EXE$GL_USRCHME,R1     ; ELSE TRY SYSTEM WIDE VECTOR
        BEQL    20$                     ; NOT PRESENT, ILLEGAL
        JSB     (R1)                    ; CALL SYSTEM WIDE USER CHME HANDLER
;
;       CALL SYSTEM-WIDE "USER" SUPPLIED PLUG-ON HANDLER FOR CHME
;       WITH UNRECOGNIZED CODES.
;
;       R0 - CODE FROM CHME/CHMK (LONGWORD)
;       R1 - ADDRESS OF ROUTINE
;       (SP) - RETURN ADDRESS TO GIVE SS$_ILLSER ERROR
;              (NORMAL RETURN IS A RET AFTER PERFORMING FUNCTION)
;
                                        ; RETURNS ONLY IF ILLEGAL CODE
20$:    BRW     ILLSER                  ;

ECASMAX=ECASCTR-1
;
;  RMS $WAIT SYNCHRONIZATION CODE.
;
;  LOOK AT FLAG IN R4 TO DETERMINE IF THIS IS A $WAIT FOR THE SAME OR DIFFERENT
;  RABS.  IF SAME, MERELY RSB;  IF DIFFERENT, WAIT ON EVENT FLAG AND THEN
;  RE-EXECUTE THE $WAIT SERVICE.
;

RMS_WAIT_SYNC:
```

```
        BLBS    R4,10$                  ;BRANCH IF DIFFERENT RABS
        RSB                             ;HANDLE WITH STANDARD STALL
10$:    TSTL    (SP)+                   ;POP RETURN PC FROM STACK
        CMPW    R0,#RMS$_STALL&^XFFFF   ;IS STALL REQUIRED?
        BEQL    20$                     ;BRANCH IF YES
        RET                             ;NO - BACK TO USER
20$:    $WAITFR_S       R3              ;WAIT ON SPECIFIED EVENT FLAG
        JMP     SYS$WAIT+2              ;RE-EXECUTE RMS $WAIT
;
;  THE FOLLOWING CODE IS AN ERROR PATH FROM THE RMS SYCHRONIZATION CODE
;  THAT PRECEDES THE RMS VECTORS.  IT WAS MOVED HERE BECAUSE CODE WAS
;  ADDED THERE AND BECAUSE THE RMS VECTORS CAN'T MOVE, THIS CODE DID.
;
;  CHECK STATUS CODE FOR ERROR OR SEVERE ERROR, IF SUCCESS THEN
;  BAD USER STRUCTURE DETECTED - RETURN ERROR IN R0, STATUS OF RECORD
;  OPERATION WILL BE LOST
;
RMS_ERR:
        BICB2   #1,RAB$B_BLN(R8)        ;CLEAR WAITING FLAG
        BLBC    R0,98$                  ;STALE SUCCESS => BAD STRUCTURE
        MOVL    #RMS$_STR,R0            ;CHANGE STATUS TO BAD STRUCTURE ERROR
98$:    BITB    #6,R0                   ;ERROR OR SEVERE ERROR?
        BEQL    99$                     ;BRANCH IF NOT
;
;  MUST RETURN TO EXEC MODE TO GENERATE POSSIBLE SYSTEM SERVICE FAILURE EXCEPTION
;
        MOVL    R0,R2                   ;STATUS CODE TO R2
        CHME    I^#SSVEXC               ;GENERATE EXCEPTION IF ENABLED
99$:    RET
.PAGE
;
;       END OF CHMK DISPATCH TABLE
;


        .PSECT  Y$CMODK,QUAD

UNIMPLEMENTED SERVICES. DEFINED TO PROVIDE CLEAN LINK.
REMOVE NAME AND VERIFY GSYSSRV ENTRY WHEN SERVICE IS IMPLEMENTED.


        CALL PER-PROCESS "USER" SUPPLIED PLUG-ON HANDLER FOR CHMK
        WITH UNRECOGNIZED CODES.

        R0 - CODE FROM CHME/CHMK (LONGWORD)
        R1 - ADDRESS OF ROUTINE
        (SP) - RETURN ADDRESS TO GIVE SS$_ILLSER ERROR
                (NORMAL RETURN IS A RET AFTER PERFORMING FUNCTION)


        JSB     EXE$LOAD_KDISP          ; CALL LOADABLE CODE DISPATCHERS

        TSTB    @#CTL$GB_SSFILTER       ; ANY INHIBIT BITS ON?
        BEQL    5$                      ; NO, ALL OKAY
        MOVZWL  #SS$_INHCHMK,R1         ; YES, SET THE EXCEPTION CODE
        BRW     INHEXCP1                ; DEAL WITH BAD CODE
```

```
5$:     MOVL    @#CTL$GL_USRCHMK,R1     ; GET PER-PROCESS VECTOR
        BEQL    10$                     ; NOT PRESENT, TRY FOR SYSTEM WIDE
        JSB     (R1)                    ; CALL PER-PROCESS HANDLER
                                        ; RETURNS ONLY IF CODE IN R0 IS NOT
;
;       CALL SYSTEM-WIDE "USER" SUPPLIED PLUG-ON HANDLER FOR CHMK
;       WITH UNRECOGNIZED CODES.
;
;       R0 - CODE FROM CHME/CHMK (LONGWORD)
;       R1 - ADDRESS OF ROUTINE
;       (SP) - RETURN ADDRESS TO GIVE SS$_ILLSER ERROR
;              (NORMAL RETURN IS A RET AFTER PERFORMING FUNCTION)
;
                                        ; HANDLED BY PER PROCESS HANDLER
10$:    MOVL    L^EXE$GL_USRCHMK,R1     ; ELSE GET SYSTEM WIDE VECTOR
        BEQL    20$                     ; NOT PRESENT, ILLEGAL SERVICE
        JSB     (R1)                    ; CALL SYSTEM WIDE HANDLER
                                        ; RETURN ONLY IF ILLEGAL CODE
20$:                                    ;
EXE$ALCDNP:                             ;
EXE$CLRPAR:                             ;
EXE$DLCDNP:                             ;

EXE$FAILURE::                           ; THIS PROCEDURE ALWAYS FAILS

        NOP
        NOP

ILLSER: MOVZWL  #SS$_ILLSER,R0          ;ILLEGAL SYSTEM SERVICE
        RET                             ;

EXE$SUCCESS::                           ; THIS PROCEDURE ALWAYS SUCCEEDS
        NOP                             ; THESE TWO INSTRUCTIONS CAN ALSO
        NOP                             ;  SERVE AS A HARMLESS ENTRY MASK
        MOVZWL  #SS$_NORMAL,R0          ; RETURN SUCCESSFUL STATUS
        RET

        .IFF    ;MPSWITCH DEFINED
        .PSECT  MPSCMOD2,BYTE
        .IFTF   ;MPSWITCH
SSFAILMAIN:                             ;SSFAIL MAIN LOGIC
        MOVL    G^CTL$GL_PCB,R1         ;GET PCB ADDRESS
        TSTW    PCB$W_MTXCNT(R1)        ;MUTEX COUNT ZERO?
        BNEQ    20$                     ;IF NEQ NO
        EXTZV   #PSL$V_CURMOD,#PSL$S_CURMOD,- ;EXTRACT PREVIOUS MODE FROM
                4(SP),-(SP)             ;SAVED PSL
        ADDL    #PCB$V_SSFEXC,(SP)      ;ADD IN BASE BIT NUMBER
        BBC     (SP)+,PCB$L_STS(R1),10$ ;IF CLEAR, FAILURE EXCEPTION DISABLED
        MOVPSL  -(SP)                   ;GET CURRENT PSL
        EXTZV   #PSL$V_CURMOD,#PSL$S_CURMOD,(SP),(SP)+ ;IF CURRENT MODE IS
        BNEQ    5$                      ;NOT KERNEL, THEN BRANCH
        SETIPL  #0                      ;FORCE IPL TO 0 FOR ERROR PATH
        .IFT
5$:     JMP     EXE$SSFAIL              ;GENERATE SYSTEM SERVICE FAILURE EXCEPTION
10$:    REI                             ;AND RETURN FROM SERVICE WITH ERROR STATUS
20$:    EXTZV   #PSL$V_IPL,#PSL$S_IPL,- ;EXTRACT PREVIOUS IPL FROM
```

```
                4(SP),-(SP)             ;SAVED PSL
        CMPL    (SP)+,#IPL$_ASTDEL      ;TEST IF AT ELEVATED IPL
        BGEQ    10$                     ;IF SO DO NOT BUGCHECK
        BUG_CHECK MTXCNTNONZ,FATAL      ;MUTEX COUNT NONZERO AT SERVICE EXIT
        .IFF    ;MPSWITCH DEFINED
5$:     IFPRIMARY <JMP G^EXE$SSFAIL>    ;IF PRIMARY, THEN CONTINUE RIGHT ALONG
                                        ;IF SECONDARY, RETURN PROCESS TO PRIMARY
        EXTZV   #PSL$V_CURMOD,#PSL$S_CURMOD,4(SP),-(SP) ;CREATE PSL WITH PREV
        ROTL    #PSL$V_PRVMOD,(SP),(SP) ; MODE CORRECT AND CURRENT MODE = KERNEL
        PUSHAB  G^EXE$SSFAIL            ;REFLECT THE EXCEPTION
        BRW     MPS$MPSCHED2            ; AND RETURN PROCESS TO PRIMARY
10$:    REI                             ;RETURN FROM SERVICE WITH ERROR STATUS
20$:    IFPRIMARY <BUG_CHECK MTXCNTNONZ,FATAL>  ;PRIMARY VERSION OF BUGCHECK
        SECBUG_CHECK MTXCNTNONZ,FATAL   ;MUTEX COUNT NONZERO AT SERVICE EXIT
        .IFT    ;MPSWITCH NOT DEFINED
;
; UPDSECW - UPDATE SECTION AND WAIT COMPOSITE SERVICE
;
        .ENABL  LSB

EXE$UPDSECW:
        CHMK    I^#UPDSEC               ;UPDATE THE SECTION
        BLBC    R0,40$                  ;BRANCH IF ERROR
        MOVL    R0,R2                   ;SAVE STATUS FROM UPDSEC

        ASSUME  UPDSEC$_EFN+4 EQ UPDSEC$_IOSB
        MOVQ    UPDSEC$_EFN(AP),-(SP)   ;PUSHL IOSB(AP), PUSHL EFN(AP)
        BRB     20$                     ;SYNCHRONIZE EFN AND IOSB
;
; COMMON WAIT CODE FOR $GETDVIW, $GETJPIW, $GETSYIW, $SNDJBCW SYSTEM SERVICES
;
; INPUTS:
;
;       R0 = STATUS FROM THE NON-WAITING VERSION OF THE SERVICE
;       EFN(AP)  = EVENT FLAG
;       IOSB(AP) = I/O STATUS BLOCK ADDRESS
;
        GETJPI_SYNCH_MASK = ^M<R2>      ;REGISTERS USED BY THIS CODE
                                        ;OTHER THAN R0 AND R1
GETJPI_SYNCH:
        BLBC    R0,40$                  ;BRANCH IF ERROR FROM ORIGINAL SERVICE
        MOVL    R0,R2                   ;SAVE STATUS FROM ORIGINAL SERVICE

        ASSUME  GETJPI$_IOSB EQ GETDVI$_IOSB
        ASSUME  GETJPI$_IOSB EQ GETSYI$_IOSB
        ASSUME  GETJPI$_IOSB EQ SNDJBC$_IOSB
        PUSHL   GETJPI$_IOSB(AP)        ;GET IOSB PARAMETER
        PUSHL   GETJPI$_EFN(AP)         ;GET EVENT FLAG PARAMETER
20$:    CALLS   #2,G^SYS$SYNCH          ;WAIT FOR EFN AND IOSB TO BE SET
        BLBC    R0,40$                  ;IF ERROR, RETURN THAT STATUS
        MOVL    R2,R0                   ;OTHERWISE RESTORE ORIGINAL STATUS
40$:    RET                             ;AND RETURN

        .DSABL  LSB
;
;       JUMPS TO REAL SYSTEM SERVICE ENTRY POINT ARE DEFINED HERE IF THE CASE
```

```
;       TABLE WON'T REACH
;
;       THESE ARE FOR USE WITHIN THIS MODULE ONLY - NOT GLOBAL ENTRY POINTS
;       ENTRY MASKS ARE PLACEHOLDERS ONLY
;
EXE$IMGACT:                             ; IMAGE ACTIVATION
        .WORD   0
        JMP     EXE$$IMGACT + 2

EXE$ASCTOID:                            ; ASCII TO IDENTIFIER CONVERSION
        .WORD   0
        JMP     EXE$$ASCTOID + 2

EXE$FINISH_RDB:                         ; FINISH RDB CONTEXT STREAM
        .WORD   0
        JMP     EXE$$FINISH_RDB + 2

EXE$IDTOASC:                            ; IDENTIFIER TO ASCII CONVERSION
        .WORD   0
        JMP     EXE$$IDTOASC + 2

;
;
        .IFTF   ;MPSWITCH
KCASMAX=KCASCTR-2

        .ENDC   ;MPSWITCH
        .ENDC   ;LIBSWITCH

        .IFTF   ;RMSSWITCH
        .IF     NDF,MPSWITCH
        .IF     NDF,LIBSWITCH
RCASMAX=RCASCTR-<1+RCASMIN>
        .ENDC
        .ENDC   ;MPSWITCH
        .IFF    ;RMSSWITCH
        .IF     NDF,MPSWITCH
        .PSECT  $$$RMSVEC,BYTE,NOWRT
        RSB                             ;NOT AN RMS EXEC MODE SERVICE
;
;  SERVICE TO MERELY MOVE RMS STATUS CODE IN R2 TO R0 AND RET,
;  THUS GENERATING A SYSTEM SERVICE FAILURE EXCEPTION IF ENABLED
;
RMS$SSVEXC=.-2
        MOVL    R2,R0                   ;MOVE STATUS CODE TO R0
        RET                             ;AND LET RET DO THE REST
        .ENDC   ;MPSWITCH
        .ENDC   ;RMSSWITCH
```

```
        .IF NDF LIBSWITCH
        .IF NDF RMSSWITCH
        .IF NDF MPSWITCH
        .SBTTL  EXE$LDB_SYNCH   -       Synchronize Loadable Services

;
; EXE$LDB_SYNCH - Synchronize Loadable Service
;
;       This routine performs a $SYNCH service in the mode of the
;       caller of a loadable service
;
;       Inputs:
;               R0      -       Main Service Status
;               (SP)    -       IOSB argument number
;               4(SP)   -       Event flag argument number
;               (FP)    -       Service Call Frame
;
;       Outputs:
;               R0      -       Status Code
;
;       Calling Sequence:
;               JMP     @#EXE$LDB_SYNCH
;
;       Returns Via:
;               RET     instruction
;

EXE$LDB_SYNCH::
        BLBC    R0,50$                          ; get out if service had error
        PUSHL   R0                              ; save service status
        CMPW    (AP),4(SP)                      ; was an IOSB specified
        BLSS    10$                             ; branch if not
        MOVL    4(SP),R0                        ; get argument offset
        PUSHL   (AP)[R0]                        ; push IOSB address
        BRB     20$

10$:    CLRL    -(SP)                           ; no IOSB so pass 0 to synch

20$:    CMPW    (AP),12(SP)                     ; was an EFN specified?
        BLSS    30$                             ; branch if not
        MOVL    12(SP),R0                       ; get argument offset
        PUSHL   (AP)[R0]                        ; push EFN number
        BRB     40$

30$:    CLRL    -(SP)                           ; no EFN so pass 0

40$:    CALLS   #2,G^SYS$SYNCH                  ; call synch system service
        MOVL    (SP)+,R0                        ; restore main service status

50$:    RET

        .ENDC   ; LIBSWITCH
        .ENDC   ; RMSSWITCH
        .ENDC   ; MPSWITCH
        .END
```

ALLOCPFN
LIS

ACCOUNT
LIS

CMODSSDSP
MAR

SYSMAR
MAR

SYSPARAM
MAR

ASTDEL
LIS